GE
Intelligent Platforms

Programmable Control Products

PACSystems* RXi

Distributed IO Controller

User's Manual, GFK-2816

December 2012

Warnings, Cautions and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note: Notes merely call attention to information that is especially significant to understanding and operating the equipment.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, nor to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Intelligent Platforms assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Intelligent Platforms makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

* indicates a trademark of GE Intelligent Platforms, Inc. and/or its affiliates. All other trademarks are the property of their respective owners.

©Copyright 2012 GE Intelligent Platforms, Inc.
All Rights Reserved

If you purchased this product through an Authorized Channel Partner, please contact the seller directly.

General Contact Information

Online technical support and GlobalCare	http://www.ge-ip.com/support
Additional information	http://www.ge-ip.com/
Solution Provider	solutionprovider.ip@ge.com

Technical Support

If you have technical problems that cannot be resolved with the information in this guide, please contact us by telephone or email, or on the web at www.ge-ip.com/support

Americas

Online Technical Support	www.ge-ip.com/support
Phone	1-800-433-2682
International Americas Direct Dial	1-780-420-2010 (if toll free 800 option is unavailable)
Technical Support Email	support.ip@ge.com
Customer Care Email	customercare.ip@ge.com
Primary language of support	English

Europe, the Middle East, and Africa

Online Technical Support	www.ge-ip.com/support
Phone	+800-1-433-2682
EMEA Direct Dial	+420-23-901-5850 (if toll free 800 option is unavailable or dialing from a mobile telephone)
Technical Support Email	support.emea.ip@ge.com
Customer Care Email	customercare.emea.ip@ge.com
Primary languages of support	English, French, German, Italian, Czech, Spanish

Asia Pacific

Online Technical Support	www.ge-ip.com/support
Phone	+86-400-820-8208
	+86-21-3217-4826 (India, Indonesia, and Pakistan)
Technical Support Email	support.cn.ip@ge.com (China)
	support.jp.ip@ge.com (Japan)
	support.in.ip@ge.com (remaining Asia customers)
Customer Care Email	customercare.apo.ip@ge.com
	customercare.cn.ip@ge.com (China)

Chapter 1. Introduction 11

1.1 Specifications 12
1.1.1 CPU Specifications 12
1.1.2 Communications Support 12
1.1.3 Mechanical Specifications 13
1.1.4 Environmental Specifications 14

1.2 RXi Controller User Features 15
1.2.1 Indicator and Port Locations 15
1.2.2 Status LED Operation 15
1.2.3 PROFINET and GbE Port LEDs 16

1.3 Additional Information 16

Chapter 2. Installation 17

2.1 Mounting Options 17

2.2 Installation Guidelines 17
2.2.1 Grounding 18
2.2.2 Mounting Orientation 19
2.2.3 Dimensions and Clearances for Installation 20

2.3 Mounting Procedures 21
2.3.1 Mounting the RXi Controller Directly on a Panel 21
2.3.2 Mounting the RXi Controller on a DIN Rail 22
2.3.3 Mounting the RXi Controller on a Panel Using a Backplate 24
2.3.4 Installing the IDM on the RXi Controller 25

2.4 Connectors and Cabling 26
2.4.1 Connecting Input Power 26
2.4.2 Connecting to the GbE Port 27
2.4.3 Connecting to a PROFINET Network 29

2.5 Replacing the RTC Battery 30

Chapter 3. Getting Started: Initial Powerup and Configuration 32

3.1 Connecting Input Power and the GbE Cable 32

3.2 Initial Powerup 32

3.3 Establishing PME Communications with the Unit and Downloading a Project 33

3.4 Configuring the Embedded PROFINET Controller (PNC) and its IO Devices on a PROFINET Network 35

3.5 Intelligent Display Module (IDM) - Basic Operations 36

Chapter 4. Configuration 37

4.1 Configuring Controller Operation 37
4.1.1 Controller Settings Parameters 38
4.1.2 Controller Scan Parameters 39
4.1.3 Controller Memory Parameters 42
4.1.4 Fault Parameters 44
4.1.5 Scan Sets Parameters 44
4.1.6 Modbus TCP Address Map 45
4.1.7 Access Control 46

4.2 Configuring the Embedded PROFINET Controller (PNC) 47
4.2.1 System Planning for PROFINET Networks 48
4.2.2 Basic Configuration Steps 48
4.2.3 Configuring the Embedded PNC 49
4.2.4 Configuring PROFINET LANs 54
4.2.5 Adding a GE Intelligent Platforms PROFINET Scanner to a LAN 56
4.2.6 Adding a Third-Party IO-Device to a LAN 59
4.2.7 Viewing / Editing IO-Device Properties 63
4.2.8 Assigning IO-Device Names 65
4.2.9 After the Configuration is Stored to the RXi CPU 67

4.3 Configuring the Embedded Ethernet Interface 68
4.3.1 Ethernet Interface Configuration Parameters 68
4.3.2 Pinging TCP/IP Ethernet Interfaces on the Network 69
4.3.3 Terminals Tab 69

4.4 Storing (Downloading) Hardware Configuration 70

Chapter 5. CPU Operation 71

5.1 CPU Sweep 71
5.1.1 Parts of the CPU Sweep 72
5.1.2 CPU Sweep Modes 74

5.2 Program Scheduling Modes 77

5.3 Window Modes 77

5.4 Data Coherency in Communications Windows 77

5.5 Run/Stop Operations 78
5.5.1 CPU Stop Modes 79
5.5.2 Stop-to-Run Mode Transition 80

5.6 Flash Memory Operation 81

5.7 Logic/Configuration Source and CPU Operating Mode at Power-up 82

5.8 Clocks and Timers 83
5.8.1 Elapsed Time Clock 83
5.8.2 Time of Day Clock 83
5.8.3 Watchdog Timer 84
5.8.4 Timed Contacts 86

5.9 System Security 87
5.9.1 Passwords and Privilege Levels 87
5.9.2 OEM Protection 88

5.10 PACSystems I/O System 89
5.10.1 I/O System Diagnostic Data Collection 89

5.11 Power-Up and Power-Down Sequences 90
5.11.1 Power-Up Sequence 90
The operation of input and output defaults for remote I/O modules in an RXi PROFINET network is described in “11.7.5, I/O Defaults Operation.“ 90
5.11.2 Power-Down Sequence 90
5.11.3 Retention of Data Memory across Power Failure 90

Chapter 6. Program Organization 91

6.1 Structure of a PACSystems Application Program 91
6.1.1 Blocks 91
6.1.2 Functions and Function Blocks 91
6.1.3 How Blocks Are Called 92
6.1.4 Nested Calls 92
6.1.5 Types of Blocks 93
6.1.6 Local Data 104
6.1.7 Parameter Passing Mechanisms 105
6.1.8 Languages 106
6.1.9 RXi Controller Instruction Set 107

6.2 Controlling Program Execution 109

6.3 Interrupt-Driven Blocks 109
6.3.1 Interrupt Handling 109
6.3.2 Configuring Timed Interrupts 110
6.3.3 Interrupt Block Scheduling 111

Chapter 7. CPU Program Data 112

7.1 Variables 112
7.1.1 Mapped Variables 112
7.1.2 Symbolic Variables 113
7.1.3 I/O Variables 113
7.1.4 Arrays 116
7.1.5 Variable Indexes and Arrays 116
7.1.6 Ensuring that a Variable Index does not Exceed the Upper Boundary of an Array 118

7.2 Reference Memory 119
7.2.1 Word (Register) References 119
7.2.2 Bit (Discrete) References 121

7.3 User Reference Size and Default 122
7.3.1 %G User References and CPU Memory Locations 122

7.4 Transitions and Overrides 123

7.5 Retentiveness of Logic and Data 123

7.6 Data Scope 124

7.7 System Status References 125
7.7.1 %S References 125
7.7.2 %SA, %SB, and %SC References 126

7.8 How Program Functions Handle Numerical Data 128
7.8.1 Data Types 128
7.8.2 Floating Point Numbers 130

7.9 User Defined Types: 132
7.9.1 Working with UDTs 132
7.9.2 UDT Properties 132
7.9.3 UDT Limits 133
7.9.4 Run Mode Store of UDTs 133
7.9.5 UDT Operational Notes 134

7.10 Operands for Instructions 134
7.11 Word-for-Word Changes 135

Chapter 8. Diagnostics 136

8.1 Fault Handling Overview: 136
8.1.1 System Response to Faults 136
8.1.2 Fault Tables 136
8.1.3 Fault Actions and Fault Action Configuration 137
8.2 Using the Fault Tables 138
8.2.1 Controller Fault Table 138
8.2.2 I/O Fault Table 140
8.3 System Handling of Faults 142
8.3.1 System Fault References 142
8.3.2 Using Fault Contacts 145
8.3.3 Using Point Faults 146
8.3.4 Using Alarm Contacts 146
8.4 Controller Fault Details 147
8.4.1 Controller Fault Groups 147
8.4.2 Controller Fault Descriptions and Corrective Actions 150
8.5 I/O Fault Details 158
8.5.1 I/O Fault Groups 158
8.5.2 I/O Fault Categories 159
8.5.3 I/O Fault Descriptions and Corrective Actions 161

Chapter 9. Gigabit Ethernet (GbE) Interface Overview and Operation 169

9.1 PACSystems GbE Communications Features 169
9.2 Station Manager 169
9.3 IP Addressing 170
9.4 SRTP Server 170
9.4.1 SRTP Inactivity Timeout 170
9.5 Modbus TCP Server Operation 171
9.5.1 Modbus Conformance Classes 171
9.5.2 Server Protocol Services 171
9.5.3 Station Manager Support 171
9.5.4 Reference Mapping 171
9.5.5 Address Configuration 173
9.5.6 Modbus Function Codes 174
9.6 Modbus TCP Client (Channels) Operation 175
9.6.2 Operation of the Communications Request 178
9.6.3 The COMM_REQ Command Block 179
9.6.4 Modbus TCP Channel Commands 180
9.7 Controlling Communications in the Ladder Program 196
9.7.1 Essential Elements of the Ladder Program 196
9.7.2 COMM_REQ Example 197
9.7.3 Troubleshooting a Ladder Program 202
9.7.4 Monitoring the Communications Channel 203
9.7.5 Sequencing Communications Requests 203
9.7.6 Major and Minor Error Codes in the COMM_REQ Status Word 204

9.8 Ethernet Interface Diagnostics 212
9.8.1 Diagnostics Tools 212
9.8.2 Troubleshooting Common Ethernet Difficulties 217
9.8.3 COMM_REQ Flooding Can Interrupt Normal Operation 219
9.8.4 Channels Operation Depends Upon Controller Input Scanning 220

Chapter 10. PROFINET Controller Overview 221

10.1 Ethernet Network Ports 221

10.2 PROFINET Networks for PACSystems 222
10.2.1 Basic System: One RXi Controller using a single port 222
10.2.2 Basic System: One RXi Controller using Multiple Ports 223
10.2.3 Basic System: Third-Party Devices and PME Programmer 224

10.3 Glossary of PROFINET Terms 225

Chapter 11. PROFINET Controller Operation 227

11.1 PROFINET Operation Overview 227
11.1.1 PROFINET Communications 227
11.1.2 Application Relationships 228
11.1.3 Types of PROFINET Communications 229
11.1.4 External Switch VLAN Priority Settings 230

11.2 Operations of the PROFINET Controller in the RXi System 231
11.2.1 Duplicate PROFINET Device IP Address 231
11.2.2 Duplicate PROFINET Controller IP Address 232

11.3 I/O Scanning 232

11.4 Data Coherency 233

11.5 Performance Factors 234

11.6 PROFINET IO Update Rate Configuration 234

11.7 RXi CPU Operations for PROFINET 235
11.7.1 Reference ID Variables for the RXi Application 235
11.7.2 PNIO_DEV_COMM Function Block 236
11.7.3 DO I/O for Remote IO Modules 237
11.7.4 Scan Set I/O for Remote I/O Modules 238
11.7.5 I/O Defaults Operation 238

11.8 PROFINET Controller Diagnostics 239
11.8.1 Powerup and Reset 239
11.8.2 PROFINET Controller Status Reporting 240
11.8.3 PROFINET IO Alarms 241
11.8.4 PROFINET Controller Faults in the Fault Tables 241

Chapter 12. PROFINET Redundant Media 242

12.1 PROFINET Media Redundancy Protocol 242

12.2 MRP Failover Performance 243
12.2.1 Guidelines for Update Rates for Bumpless Operation during MRP Ring Recovery 243
12.2.2 Effect of MRC LinkUp/LinkDown Detection on Failover Performance 244
12.2.3 Test Packet Timeout Interval 244

12.3 Ring Topology with One Controller 245
12.4 Ring Topology with Multiple Controllers 246
12.5 Setting Up Media Redundancy Protocol 247
12.5.1 Media Redundancy Setup for a PROFINET Controller 247
12.5.2 Sequence for Enabling Media Redundancy 247
12.6 Sequence for Replacing a Media Redundancy Manager 248
12.7 Procedure for Disabling Media Redundancy 248

Chapter 13. PROFINET Network Management 249

13.1 SNMP 249
13.1.1 Overview of SNMP 249
13.1.2 Supported SNMP Features 250
13.1.3 SNMP Read Access 251
13.1.4 SNMP Write Access 251
13.1.5 MIB-II Groups Supported 251
13.1.6 MIB-II System Group Values 252
13.2 LLDP 253
13.2.1 Overview of LLDP 253
13.2.2 LLDP Operation 253
13.2.3 LLDP TLVs 253

Appendix A CPU Performance Data 256

Appendix B PROFINET IO Performance Examples 264

B.1.1. 1ms PROFINET Update Rate Systems 265
B.1.2. 16ms PROFINET Update Rate Systems 266

Appendix C User Memory Allocation 267

C.2.1. %L and %P Program Memory 268
C.2.2. Program Logic and Overhead 268

Appendix D Product Certifications and Standards 269

Chapter 1. Introduction

The RXi Controller combines the PACSystems control engine with PROFINET-based distributed I/O and gigabit Ethernet (GbE) communications into a single package.

- **Dual-core COMExpress CPU architecture** –provides high performance in rugged applications.
- **Integrated redundant PROFINET I/O interface** provides a Gigabit/100 Megabit Ethernet I/O network connection with built-in Media Redundancy Protocol (MRP) delivering IO cabling redundancy with no external switches.
- **Built-in data storage** – Internal industrial grade SSD drive provides long-term data retention.
- **High performance industrial control platform** incorporates patented thermal monitoring technology and sophisticated passive cooling techniques that eliminate reliance on fans for cooling.
- **Embedded Ethernet interface** supports up to 32 simultaneous SRTP Server connections, up to 16 simultaneous Modbus TCP Server connections, and up to 16 simultaneous Modbus TCP Client channels.

An Intelligent Display Module with a multi-touch panel display is mounted on the RXi Controller for enhanced operator usability.

RXi Controller (shown with Intelligent Display Module)

1.1 Specifications

Part Numbers

ICRXICTL000	RXi Controller
ICRXIACCIDM01	Intelligent Display Module (IDM)
ICRXIACCBPL (rev B or later required)	Optional Backplate for DIN rail mounting
IC690ACC001	Real Time Clock (RTC) battery, replacement

1.1.1 CPU Specifications

Processor	1.0GHz
Program storage	10 MB of user memory and 10MB of user flash storage
Non-volatile storage	Non-volatile storage (NVS) can retain data indefinitely without loss of data integrity.
Time of day clock (RTC) accuracy	Maximum drift of ±2 seconds/day at 25°C
Real Time Clock battery	Estimated life of 5 years; must be replaced every 5 years on a regular maintenance schedule.
Elapsed time clock (internal timing) accuracy	± 0.01% maximum
Floating point	64 bit
Program Blocks	Up to 512 program blocks. Maximum size for a block is 128KB.
Memory	%I and %Q: 32Kbits for discrete %AI and %AQ: configurable up to 32Kwords %W: configurable up to the maximum available user memory Symbolic: configurable up to the maximum available user memory

1.1.2 Communications Support

Gigabit Ethernet Interface Specifications

Port connector	RJ-45 Port: 8-pin female shielded RJ-45 with 10/100/1000 Mbps
LAN	IEEE 802.2 Logical Link Control Class I IEEE 802.3 CSMA/CD Medium Access Control 10/100 Mbps
Number of IP addresses	One
Max. no. of SRTP server connections	Forty-seven SRTP connections available through the front panel GbE port
Modbus TCP server connections	16
Modbus TCP/IP channels (client)	16
Remote Station Manager over UDP	Yes - Monitor mode commands only. Refer to the *Station Manager Manual,* GFK-2225 for command operation.
Configurable Advanced User Parameters	Not supported in initial release.

PROFINET Controller (PNC) Specifications

PROFINET version	PROFINET Version 2.2 General Class A IO-Controller
Port connectors	Two RJ-45 with 10/100/1000 Mbps. Supports PROFINET and Media Redundancy Protocol (MRP).
LAN	IEEE 802.2 Logical Link Control Class I IEEE 802.3 CSMA/CD Medium Access Control 10/100/1000 Mbps
Maximum I/O Memory	128 Kbytes of combined input/output memory
CPU status bits	32
PROFINET IO-Device data update rates on the PROFINET network	Configurable: 1ms, 2ms, 4ms, 8ms, 16ms, 32ms, 64ms, 128ms, 256ms and 512ms
Number of IP addresses	One
Number of MAC addresses	Seven. One per external port and four internal.
System maximum limits	
IO-Devices per IO-Controller	128 per PNC at 16–512ms update intervals. For limits at shorter update intervals, refer to “4.2.1, System Planning.”
IO-Devices per network	255 per network
IO-Devices per RXi CPU	128 per RXi Controller
IO-Controllers per network	8
Number of PROFINET slots per device	256
Number of PROFINET subslots per slot	256
Number of PROFINET submodules per RXi CPU	2048
Programmer Limits	
PROFINET IO networks	16
Number of IO-Controllers	128
Number of IO-Devices	4080 (255 per network × 16 PROFINET networks)
Total number of devices	4208

1.1.3 Mechanical Specifications

Dimensions: RXi Controller with lid Backplate	 191.8mm x 115.6mm x 78.7mm (7.55 in x 4.55 in. x 3.1 in) 223.5 mm x 115.6 mm x 17.8 mm (8.80 in. x 4.55 in. x 0.7 in.)
Weight: RXi Controller with lid Backplate	 1.814 Kg (4 lbs) 0.908 Kg (2 lb)
Power requirements (RXi Controller with IDM)	1.5 A at 24 VDC (18–32 VDC) = 36 Watts

1.1.4 Environmental Specifications

Note: The RXi Controller shall be installed in a location that is not exposed to corrosive gases or liquids, rain, or direct sunlight, and that meets the environmental specifications listed below.

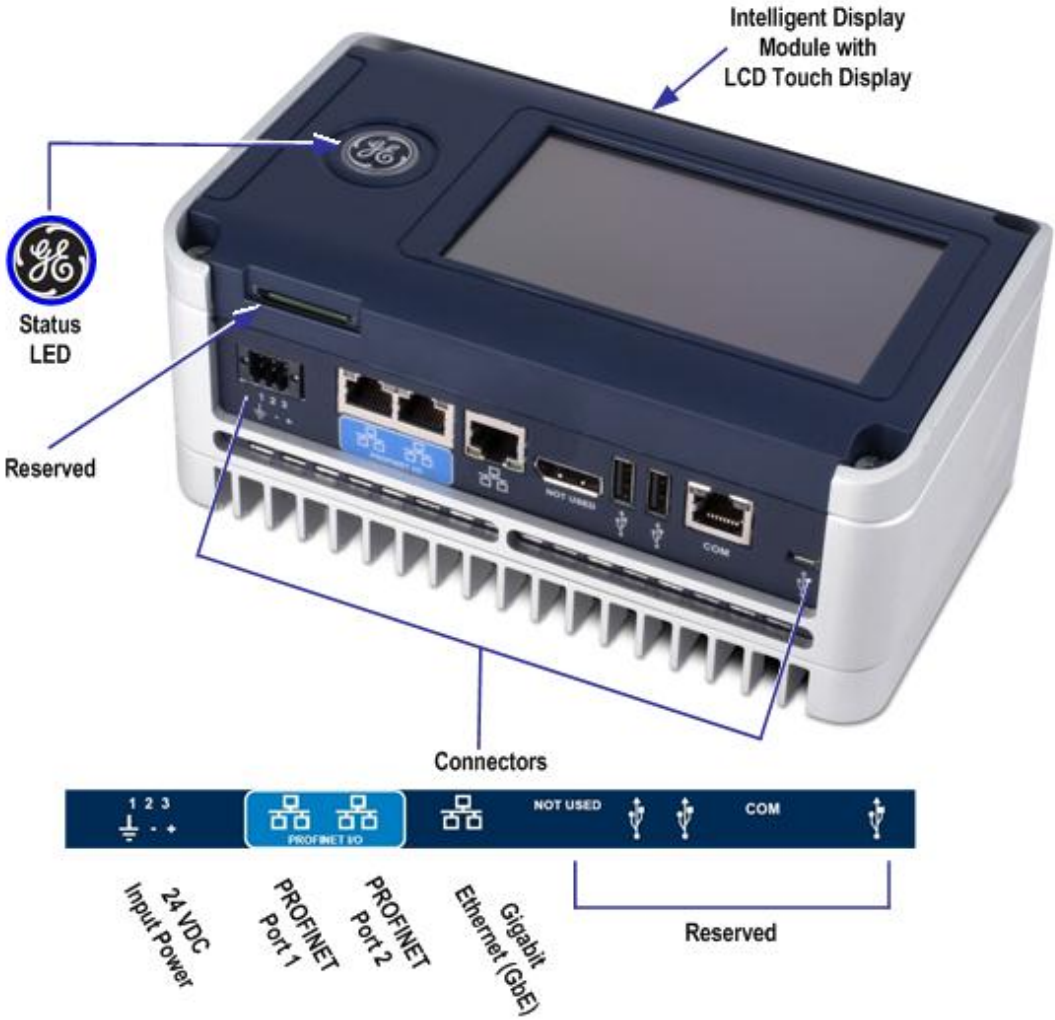
Description	*Comments*
Vibration	10 - 57 Hz, 0.006 in. displacement peak-peak 57 - 500 Hz, 1.0 g acceleration
Shock	15G, 11ms
Ambient Operating Temperature	Panel mount: -25° to +50°C: [inlet] (-13° F to 122°F) DIN rail mount: -25° to +45°C: [inlet] (-13° F to 113°F)
Storage Temperature	-40° to +85° C (-40°F to 185°F)
Humidity	5% to 95%, non-condensing
Environment	Pollution Degree 2 as defined below.
Altitude	0–2000 m

A ***Pollution Degree 2 environment***:

- Pollution Degree 2 applies where there is only non-conductive pollution that might temporarily become conductive due to occasional condensation.

1.2 RXi Controller User Features

1.2.1 Indicator and Port Locations

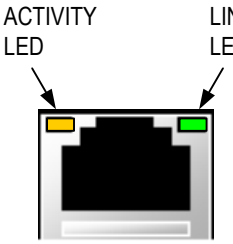


1.2.2 Status LED Operation

LED State	RXi Controller State
Blue, blinking	Power on
Green, blinking	Operating system initialization complete/Jump to Application
Blue, solid	Stop mode
Green, solid	Run mode
Red, solid	Stop Fault
Red, steady blink	Stop-Halt
Red, blink code	Critical Failure

1.2.3 PROFINET and GbE Port LEDs

Each port has two LED indicators, **ACTIVITY** and **LINK**.



LED	LED State	Operating State
ACTIVITY	Blinking, Amber	10/100/1000 activity
LINK	On, Green	Gigabit link status

1.3 Additional Information

For additional information, refer to the manuals listed below. Manuals can be downloaded from the Support website.

VersaMax PROFINET Scanner User's Manual, GFK-2721

PACSystems RSTi Network Adapter Manual, GFK-2746

VersaPoint IC220PNS001 Datasheet, GFK-2572

VersaPoint IC220PNS002 Datasheet, GFK-2573

VersaPoint System Installation Manual, GFK-2736

C Programmer's Toolkit for PACSystems User's Manual, GFK-2259E

Chapter 2. Installation

2.1 Mounting Options

Note: Before selecting a mounting method, refer to the thermal requirements in "1.1.4, Environmental Specifications."

- Direct mounting onto a panel.
- DIN rail mounting using the optional Backplate (ICRXIACCBPL).
- Panel mounting using the optional Backplate.

Mounting methods are described in detail on pages 21 through 25.

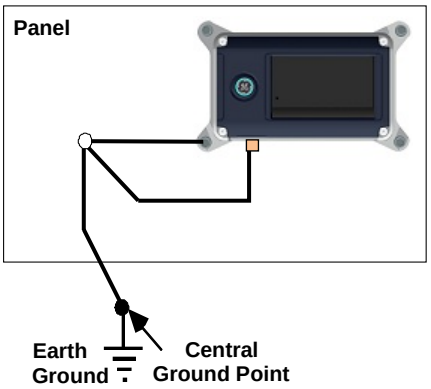
2.2 Installation Guidelines

- The RXi Controller must be mounted with its cooling fins vertical to ensure adequate airflow.
- The panel should be capable of supporting the weight of the controller without distortion to the panel.
- Adequate airflow around the exterior of the unit is essential to maintain safe interior temperature of the unit. Inlets and outlets must not be obstructed. For details, see page 20.
- You may need to allow more space for installation of cables and connectors than what is required for heat dissipation. To avoid impacting mechanical reliability and signal quality, cable installation must comply with the minimum bend radius specified by the cable manufacturer.

2.2.1 Grounding

Note: These grounding connections serve as a path for reducing electrical interference and emissions and are required for the RXi Controller to comply with the standards identified in Appendix D.

- The panel that the controller or DIN rail is mounted to must have a safety ground connection to protective earth. This ground wire must be at least #16 AWG (1.31mm^2).
- If the controller is mounted on the Backplate, connect the ground wire to the 8mm grounding stud on the Backplate and to protective earth (ground).
- Connect the frame ground connection on the power plug to protective earth.
- Terminate all ground wires at the same grounding point.
- Make all ground wires as short as possible.



Ground Wiring Example

2.2.2 Mounting Orientation

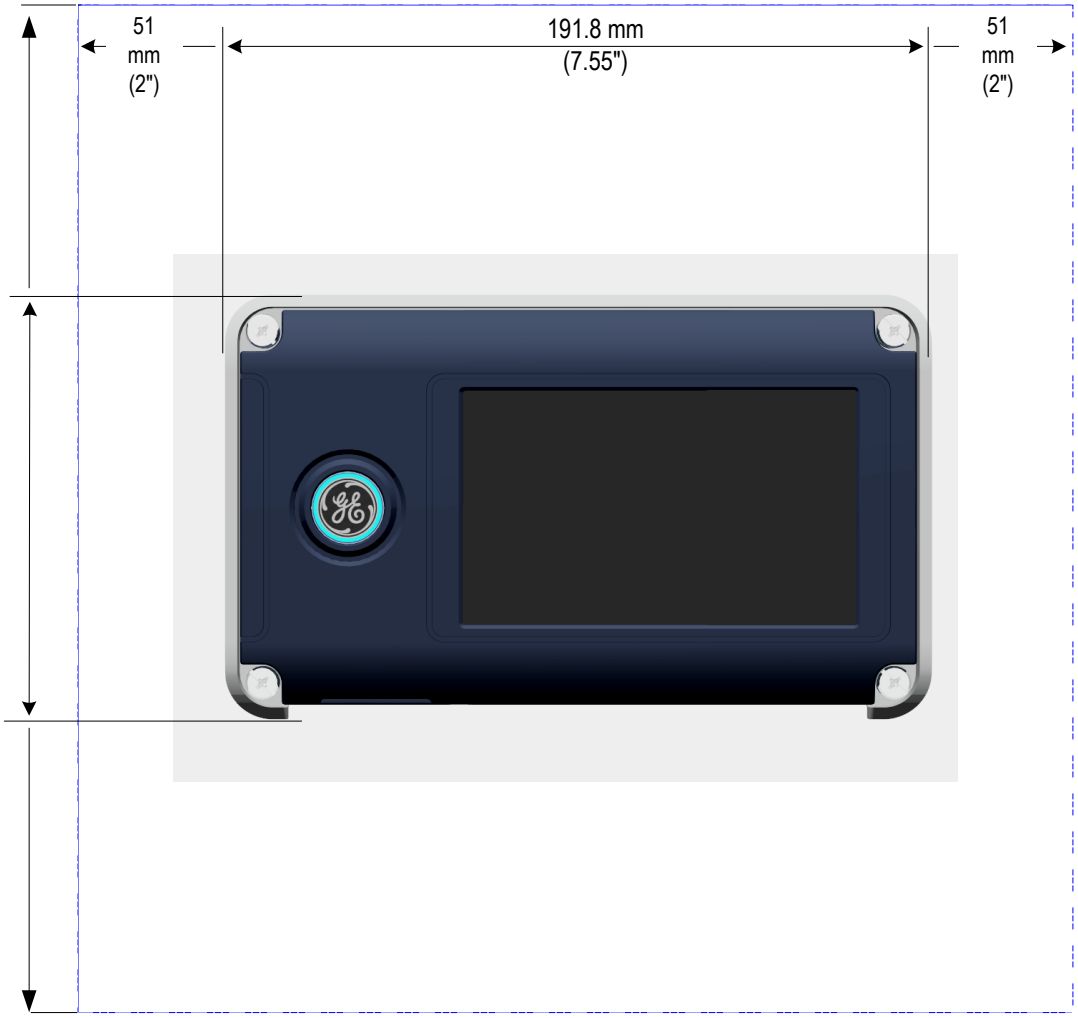
The cooling fins on the back of the controller must be vertical.

Correct

Incorrect

2.2.3 Dimensions and Clearances for Installation

Dimensions	RXi Controller:	191.8 mm x 115.6 mm x 81.3 mm (7.55 in x 4.55 in. x 3.2 in)
	Backplate:	226 mm x 137 mm x 12 mm (8.90 in. x 5.39 in. x 0.47 in.)
Minimum clearances for heat dissipation	Each side: 51mm (2 inches) Top and bottom: 127mm (5 inches)	



Note: Dimensions for clearances *not* shown to scale.

Clearances for Heat Dissipation

2.3 Mounting Procedures

The RXi Controller has four captive machine screws in its base for attaching the unit to the Backplate or panel.

Note: Before selecting a mounting method, refer to the thermal requirements in "1.1.4, Environmental Specifications."

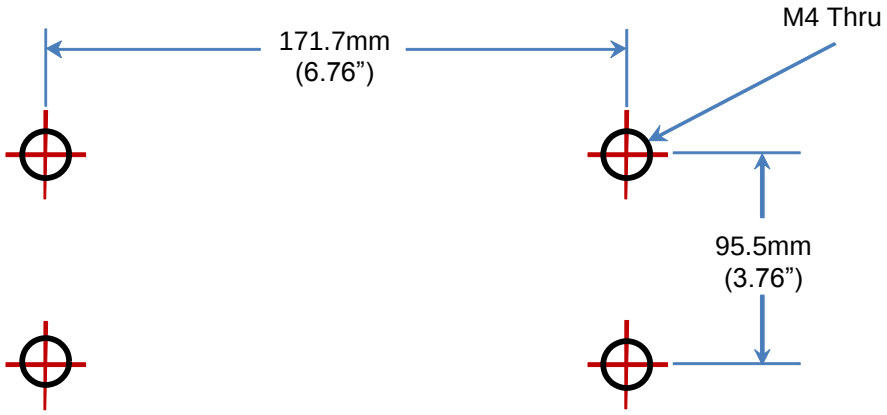
2.3.1 Mounting the RXi Controller Directly on a Panel

You will need the following:

- One flathead or large Phillips screwdriver
- One 2.5mm hex driver (included with the RXi Controller)
- #29 (0.116") drill bit and M4x0.7 tap

The Controller has four captive machine screws in its base for attaching the unit to the panel.

1. Drill four mounting holes using the spacing shown in the following drawing and tap M4x0.7 thru.
2. Align the Controller's four mounting screws with the mounting holes in the panel.
3. Using the hex driver, hand-tighten the mounting screws.
4. To install the IDM, see 2.3.4, "Installing the IDM on the RXi Controller."



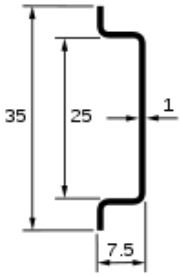
Drilling Pattern for Direct Panel Mounting

2.3.2 Mounting the RXi Controller on a DIN Rail

You will need the following:

- One 2.5mm hex driver (included with the RXi Controller)
- One flathead or large Phillips screwdriver
- One Backplate, ICRXIACCBPL

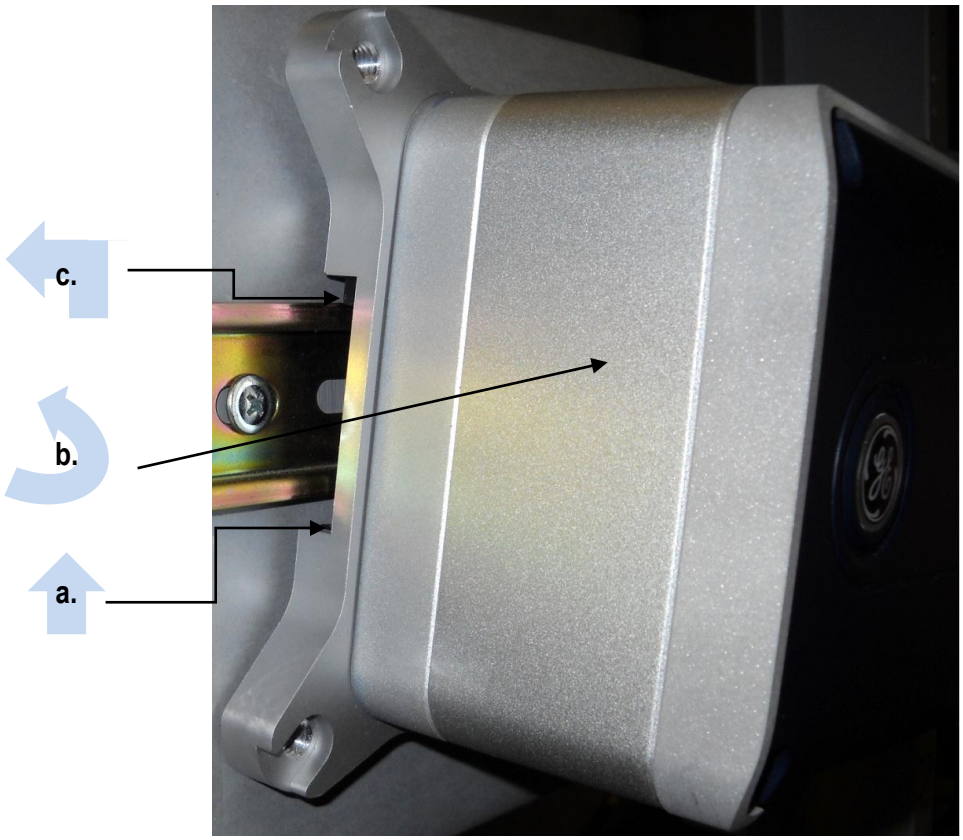
When installed on the Backplate, the controller mounts to a standard EN 50022 DIN rail with the following dimensions.



1. Install the RXi Controller on the Backplate.
   a. Place the RXi and Backplate on a flat surface.
   b. Make sure the top of the RXi is positioned at the top of the Backplate.
   c. Align the RXi's four mounting screws with the mounting holes in the Backplate.
   d. Using the 2.5mm hex driver (provided), tighten the four mounting screws.

   e. To install the IDM, see 2.3.4, "Installing the IDM on the RXi Controller."

2. Install the unit on the DIN Rail.
   a. Start with the unit rotated approximately 20º from the panel.
   b. Engage the latches at either the top or bottom of the black DIN rail clip (attached to the adaptor) onto the DIN rail.
   c. Rotate the unit toward the DIN rail and engage both sets of latches with the rail.
   d. Press the unit into place.



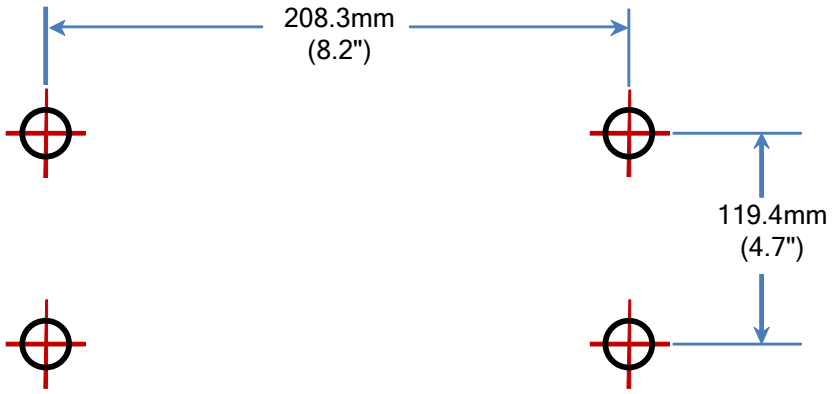
2.3.3 Mounting the RXi Controller on a Panel Using a Backplate

When mounting the Controller/Backplate assembly on the panel, you can insert the screws from the front of the Backplate or from the back of the panel.

You will need the following:

- One ICRXIACCBPL Backplate
- Four machine screws with maximum thread size ¼-20 (standard) or M6 (metric)
- If you want to insert the screws from the front of the panel, you will need four nuts. (If the screws are inserted from the back, they are threaded into the Backplate holes.)
- Driver for the selected machine screws

1. Install the controller on the Backplate. (See step 1 on page 21.)
2. Drill four holes in the panel using the spacing shown in the following drawing.



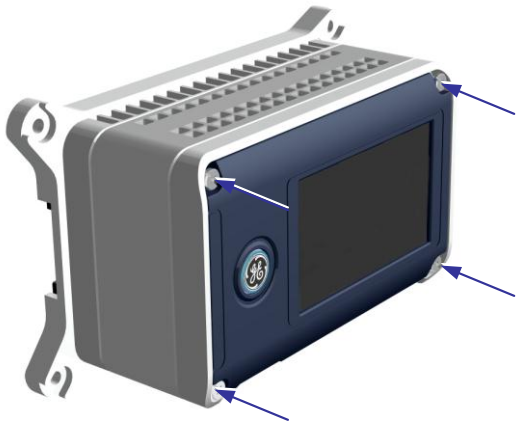
Drilling Pattern for Panel Mounting Using a Backplate

3. Remove the rubber feet from the Backplate mounting holes.
4. Use the four mounting screws to attach the Controller/Backplate assembly to the panel. Torque settings for the fasteners used to mount the Backplate should be determined according to standard industry practices or manufacturer's recommendations.

2.3.4 Installing the IDM on the RXi Controller

The IDM should be installed after the RXi Controller has been mounted on the panel or DIN rail.

Place the IDM on the RXi and use a flathead or large Phillips screwdriver to hand-tighten the four captive screws.



2.4 Connectors and Cabling

Power and communication connectors for the RXi Controller are described in this section.

All connectors are located on the bottom panel of the controller.

Summary of Cabled Ports

Connection	Maximum Length	Shielding Required
24 VDC	3 m (9.8 ft.)	No
PROFINET 1	100 m (328 ft.)	Yes[1]
PROFINET 2	100 m (328 ft.)	Yes[1]
Gigabit Ethernet (GbE)	100 m (328 ft.)	Yes[1]

[1] Shielded cable required for 1 Gbps operation.

2.4.1 Connecting Input Power

You will need:

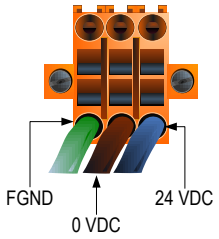
- An 18–32VDC, 36W (minimum) power supply. The power supply must be either:

  A UL Listed (or equivalent) Limited Power Source (LPS) or Class 2 power source capable of providing 18–32VDC, 36W minimum

  or

  A low voltage/limited current (LVLC) power source (the combination of an isolated DC supply and a fuse, listed 32VDC minimum and 3A maximum, connected in series with the output of the power supply).
- A power cord with 18 AWG wires (0.82mm^2)
- A frame ground wire, 18 AWG (0.82mm^2)
- An input power terminal block (provided)

1. Using the power cord, attach the power supply to the power terminal block.
2. Recommended wire stripping length is 7mm (0.28 in).
3. Insert the plug into the RXi's Input Power connector and securely tighten the attaching screws.



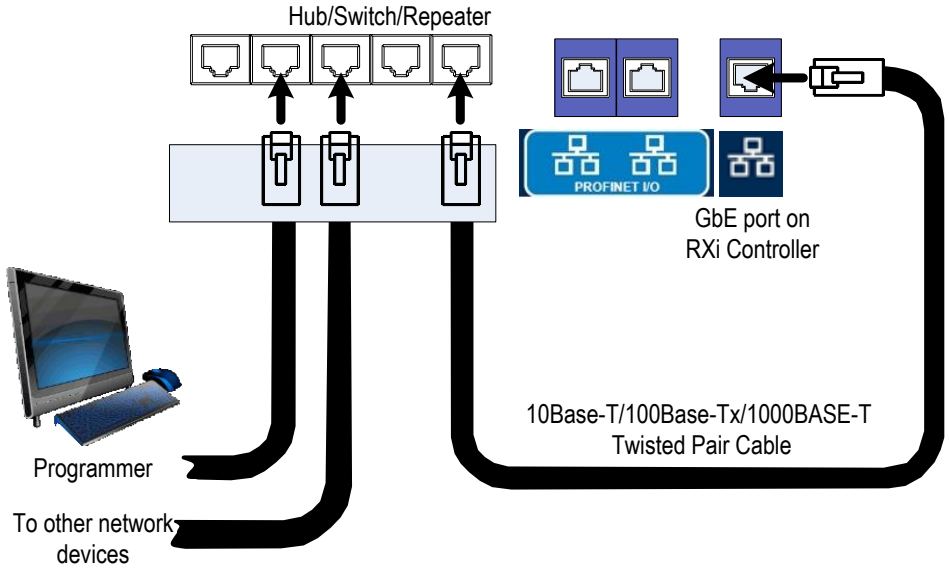
Note: There are no user-serviceable fuses in the RXi Controller.

2.4.2 Connecting to the GbE Port

The RXi Controller provides an RJ-45 Ethernet network port connector that supports 10BASE-T, 100BASE-TX and 1000BASE-T communications. The port automatically senses the data rate (10, 100 or 1000Mbps), duplex (half duplex or full duplex), and cabling arrangement (straight through or crossover) of the attached link.

The following diagram shows a typical connection of the PC running Proficy* Machine Edition (PME) software to the RXi's GbE port.

Note: Shielded cable is required for 1 Gbps operation.



2.4.2.1 Ethernet Media

The RXi Controller can operate directly on 10BASE-T, 100BASE-TX or 1000BASE-T media via its network port. All three arrangements can use up to 100m of twisted pair cable between each node and a switch, hub, or repeater.

Note: For all three types, shielded twisted pair (STP) cable is required to maintain CE compliance.

10BASE-T: Two pairs of wire are used, one for transmission, and the other for receive.

100BASE-TX: Two pairs of wire are used, one for transmission, and the other for receive.

1000BASE-T: Four pairs of wire are used for simultaneous transmission and receive in both directions.

Note: Pin assignments are provided for diagnostic purposes only. Ethernet cables are available from commercial distributors. We recommend purchasing rather than making cables.

10Base-T/100Base-Tx Port Pin Assignments

Pin Number	Signal	Description
1	TD+	Transmit Data +
2	TD-	Transmit Data -
3	RD+	Receive Data +
4	NC	No connection
5	NC	No connection
6	RD-	Receive Data -
7	NC	No connection
8	NC	No connection

1000Base-T Port Pin Assignments

Pin Number	Signal	Description
1	BI_DA+	Bi-directional pair A+ (Transmit)
2	BI_DA-	Bi-directional pair A- (Transmit)
3	BI_DB+	Bi-directional pair B+ (Receive)
4	BI_DC+	Bi-directional pair C+
5	BI_DC-	Bi-directional pair C-
6	BI_DB-	Bi-directional pair B- (Receive)
7	BI_DD+	Bi-directional pair D+
8	BI_DD-	Bi-directional pair D-

The operation of the Ethernet port LED indicators, **ACTIVITY** and **LINK**, is described in 1.2.3, “PROFINET and GbE Port LEDs.”

2.4.3 Connecting to a PROFINET Network

The RXi Controller provides two RJ-45 ports for PROFINET network connections. They can also be used for general Ethernet communications on a 10BaseT, 100BaseTX, or 1000Base-T IEEE 802.3 network.

PROFINET IO can be connected to either of the two PROFINET ports. Use of Media Redundancy Protocol (MRP) requires both PROFINET ports. If MRP is not used, the PROFINET IO can be connected to either port.

Note: For sample PROFINET network topologies, see 10.2, PROFINET Networks for PACSystems.

Caution

Do not connect both ports on the RXi to the same device, either directly or indirectly, unless Media Redundancy is enabled in the PROFINET interface configuration.

If Media Redundancy will be used, do not close the network ring until a Media Redundancy configuration that contains a Media Redundancy Manager (MRM) node has been downloaded to the RXi. If an MRM is not present, packets can continuously cycle on the network, using up significant network bandwidth.

2.4.3.1 PROFINET Media

10BaseT: uses a twisted pair cable of up to 100 meters in length between a node and another node, switch, hub, or repeater. 10Mbs can be used for general Ethernet traffic, but not for PROFINET communications.

100BaseTX: uses a cable of up to 100 meters in length between a node and another node, switch, hub, or repeater. The cable should be data grade Category 5e or better shielded twisted pair (STP).

1000BaseT: uses a cable of up to 100 meters in length between a node and another node, switch, hub, or repeater. The cable should be data grade Category 5e or better STP.

The operation of the Ethernet port LED indicators, **ACTIVITY** and **LINK**, is described in 1.2.3, “PROFINET and GbE Port LEDs.”

2.5 Replacing the RTC Battery

The real-time clock is backed up by a lithium coin cell battery, IC690ACC001, which has an estimated life of 5 years and must be replaced every 5 years on a regular maintenance schedule. There are no diagnostics or indicators that monitor RTC battery status.

If the RTC battery fails, the CPU date and time is reset to 12:00 AM, 01-01-2008 at startup. The CPU operates with a failed or missing RTC battery; however the initial CPU TOD clock information will be incorrect.

Warning

If the unit has been recently powered down, the battery may be too hot to touch safely. To avoid burns, use precautions, such as allowing the battery sufficient time to cool before handling it.

Caution

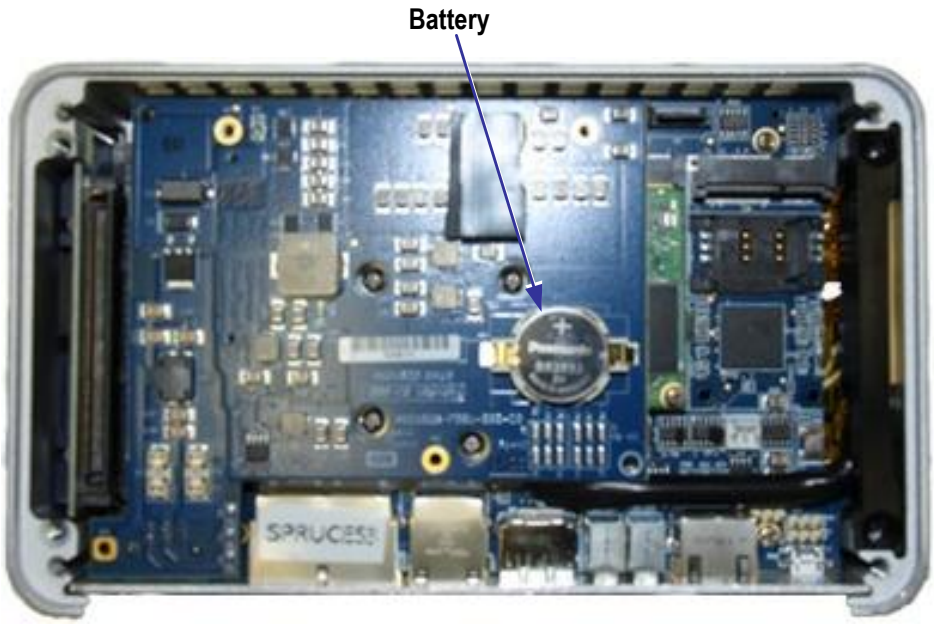
To avoid damage from electrostatic discharge, adhere to the following precautions when removing the Display Module:

- **Wear a properly functioning antistatic strap and be sure that you are fully grounded. Never touch any components inside the Display Module unless you are wearing an antistatic strap.**
- **Extra caution should be taken in cold, dry weather, when static charges can easily build up.**

You will need the following:

- One flat (or large Phillips) screwdriver
- One small Phillips screwdriver
- IC690ACC001 battery

1. Remove power from the RXi Controller.
2. Loosen the four captive screws on the Display Module and remove it.
3. Remove the inner lid from the RXi Controller.
4. Remove the RTC battery from the retaining clip, being careful to not bend the positive terminal clip.



Warnings

Use of a different type of battery than that specified here may present a risk of fire or explosion.

Battery may explode if mistreated. Do not recharge, disassemble, heat above 100°C (212°F) or incinerate.

Note: For proper disposal, refer to Battery Disposal document 82A1540-MD01.

5. Install the new battery in the retaining clip.

   Install the battery with the positive (+) side up (with the + side away from the circuit board). The coin cell must be inserted at an angle under the positive terminal clip and then slid into the carrier and snapped into place.
6. Replace the inner lid on the RXi Controller and hand-tighten the four captive screws to secure it.
7. Replace the Display Module on the RXi Controller and hand-tighten the four screws to secure it.

Chapter 3. Getting Started: Initial Powerup and Configuration

Before you attempt to power up the RXi Controller for the first time, inspect the unit for loose or damaged components. If damage is observed (for example, in the form of bent component leads or loose components), contact GE Intelligent Platforms for additional instructions. Depending on the severity of the damage, it may be necessary to return the product to the factory for repair.

Do not apply power to the unit if it has visible damage. Applying power to a unit with damaged components may cause additional damage.

You will need the following:

- Power supply and power cord. See 2.4.1, “Connecting Input Power” for requirements.
- Ethernet cables. See 2.4.2, “Connecting to the GbE Port” for requirements.
- Proficy Machine Edition software, version 7.50 or later

3.1 Connecting Input Power and the GbE Cable

1. Connect input power as described in 2.4.1, “Connecting Input Power.”
2. Connect the PC running PME software to the GbE port as described in 2.4.2, “Connecting to the GbE Port.”

3.2 Initial Powerup

1. Apply Power and start the RXi Controller
2. When 24 VDC is applied to the input terminals, the powerup process begins. During startup and initialization, the Status LED blinks blue and then green.

- ***Blinking blue*** while the RXi Controller is starting up
- ***Solid blue*** when the RXi Controller has completed startup. (It will be in Stop mode).

  Note that the IDM identifies the RXi Controller as “Default” until a project is downloaded to it.

3.3 Establishing PME Communications with the Unit and Downloading a Project

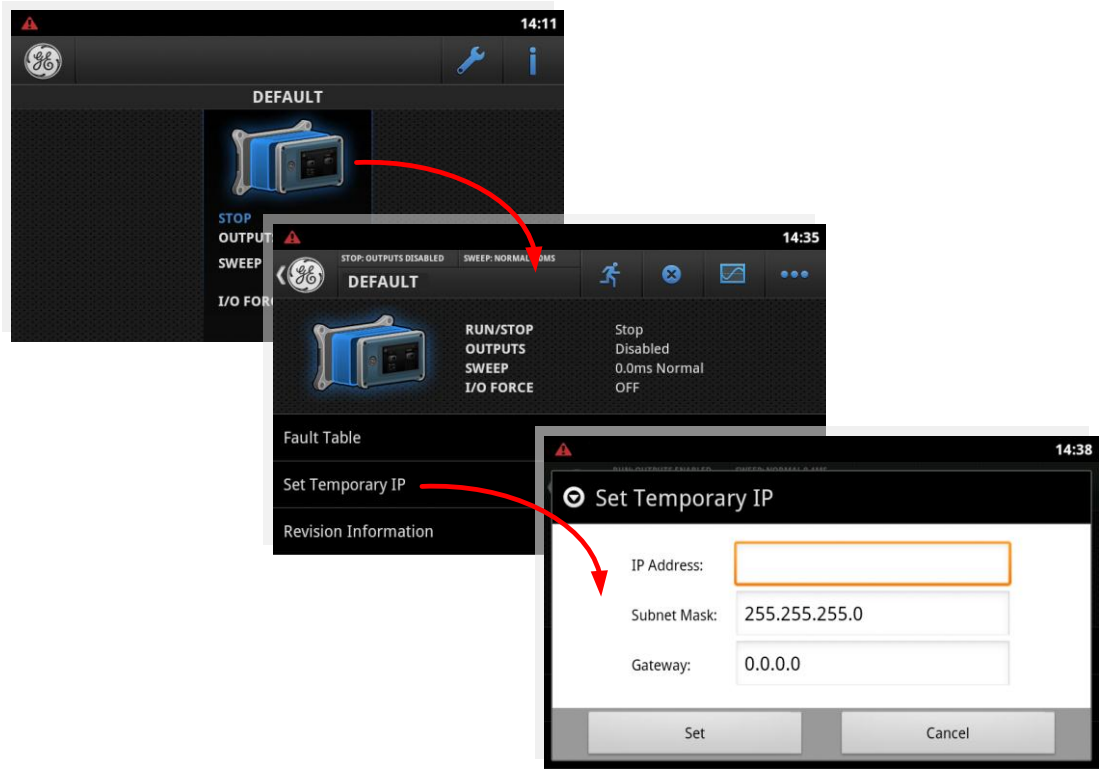
1. Set the IP Address on the RXi.

Caution

The Ethernet interface must be configured with the correct IP Address for proper operation in a TCP/IP Ethernet network. Using incorrect IP addresses can disrupt network operation for the RXi and other nodes on the network.

- The GbE port's default IP address is 192.168.0.100.
- The subnet and subnet mask of the RXi and the PME computer must match.
- The subnet for the GbE port and the PROFINET ports must be different.

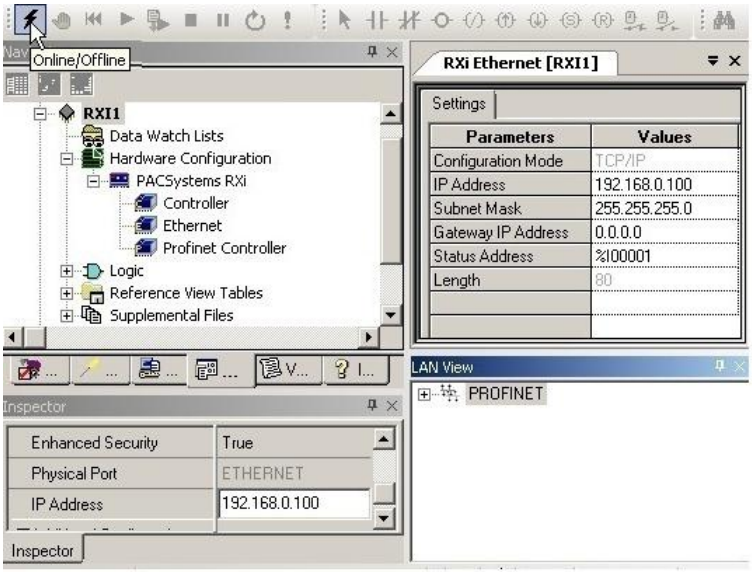
If the default IP address is not suitable to your application, you can use the IDM to assign a temporary IP address to the controller. To enter a new IP address using the IDM, first perform a Clear All operation, then change the IP address using the Set Temporary IP function.



Setting the Temporary IP Address Using the IDM

2. Go online with PME software.

In PME, create a project with a PACSystems RXi target.
In the Inspector, set the active target's IP Address property to the RXi's address.
Right click the target and select Go Online or click the Online/Offline toolbar button.



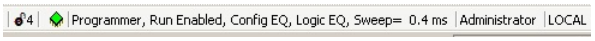
Sample Configuration for RXi Ethernet Interface

3. Download a project and start the RXi. From the RXi target right-click menu, or the PME toolbar:

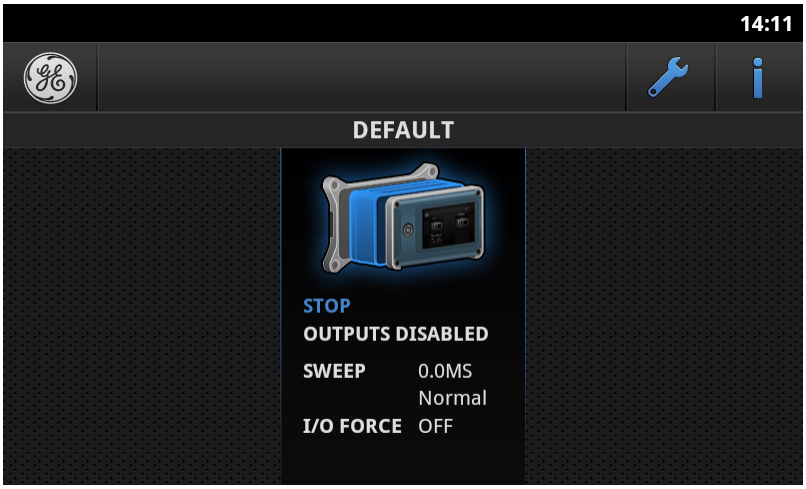
Set PME to Monitor mode.

Select Download and Start.

The PME status bar indicates the status of the RXi Controller.



Note that the IDM identifies the RXi Controller as "Default" until a project is downloaded to it.

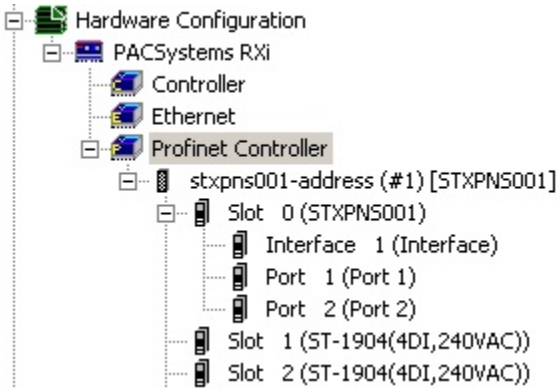


IDM Display after Initial Powerup

3.4 Configuring the Embedded PROFINET Controller (PNC) and its IO Devices on a PROFINET Network

Basic configuration steps for a PROFINET network:

1. Configure the PROFINET Controller parameters and properties in PME's hardware configuration tool.
2. Add IO-Devices to the PROFINET network. These can be GE Intelligent Platforms PROFINET Scanner (PNS) modules or third-party IO-Devices.
3. Configure the IO-Devices.
4. Store the configuration data from PME to the RXi.



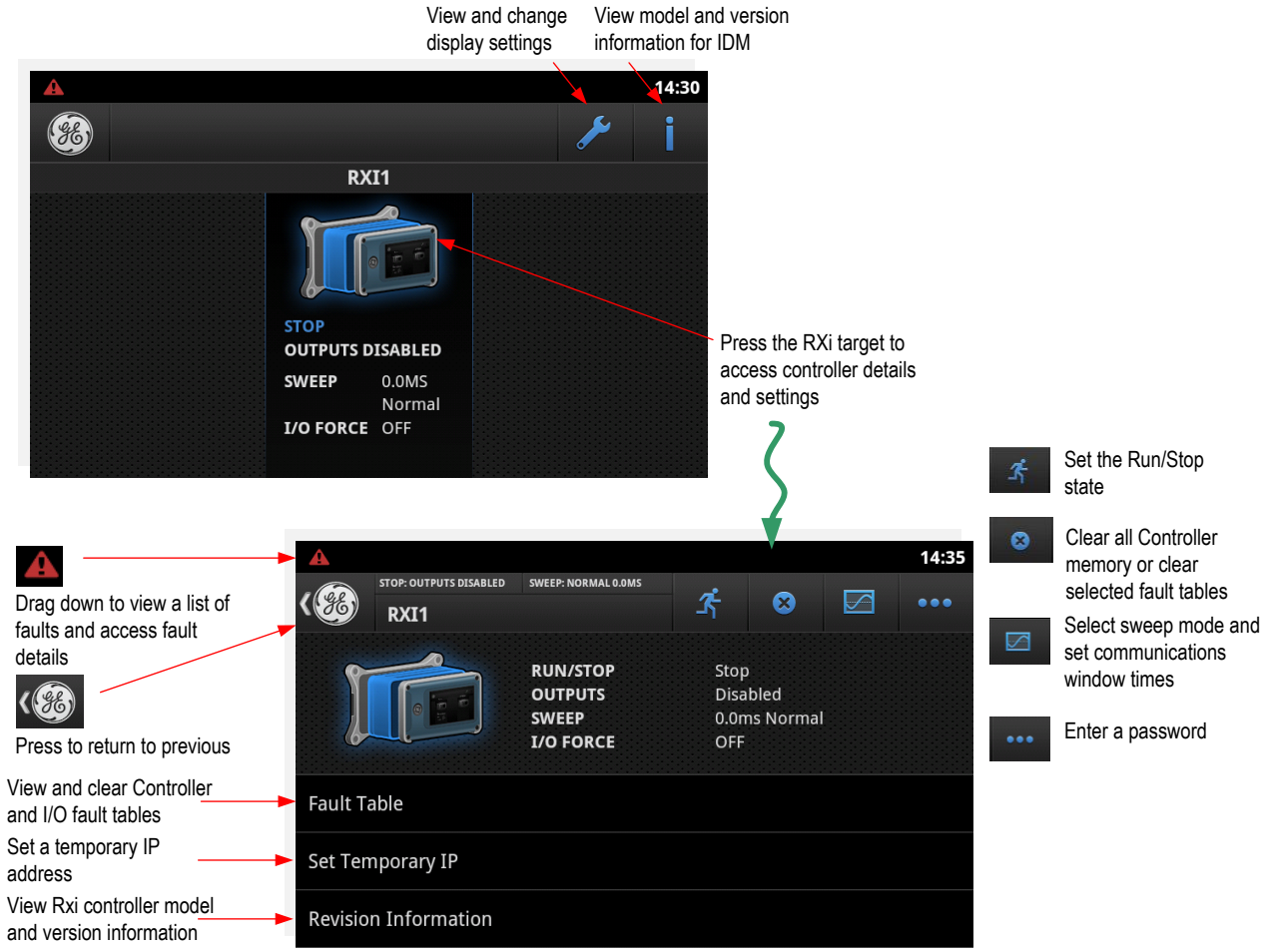
Sample PROFINET Network Configuration with RSTi PNS Node

For additional information about setting up your PROFINET network, refer to 4.2, "Configuring the Embedded PROFINET Controller."

Note: The subnet for the two PROFINET ports ***must be different*** from the subnet for the port used for the PME software connection. Until a configuration is stored for the PROFINET ports' IP address, subnet and gateway, their default values are 0.0.0.0. Also, the device name of the PROFINET Controller functionality is NULL (empty string).

3.5 Intelligent Display Module (IDM) - Basic Operations

Although PME is required to create and download configuration and logic to the RXi, you can perform many operations with the IDM. These include setting an initial IP address, starting or stopping the RXi, and viewing and clearing fault tables.With the IDM, you can perform many operations on the RXi Controller without the need for going online with the programming software.


View and change display settings
View model and version information for IDM
14:30
RXI1
STOP
OUTPUTS DISABLED
SWEEP 0.0MS Normal
I/O FORCE OFF
Press the RXi target to access controller details and settings
Set the Run/Stop state
Clear all Controller memory or clear selected fault tables
Select sweep mode and set communications window times
Enter a password
Drag down to view a list of faults and access fault details
Press to return to previous
14:35
STOP: OUTPUTS DISABLED
SWEEP: NORMAL 0.0MS
RXI1
RUN/STOP Stop
OUTPUTS Disabled
SWEEP 0.0ms Normal
I/O FORCE OFF
View and clear Controller and I/O fault tables
Set a temporary IP address
View Rxi controller model and version information
Fault Table
Set Temporary IP
Revision Information

The use of IDM's "Provide Password" allows entries that match the passwords defined using PME.

CAUTION

Passwords loaded to Flash cannot be cleared using Clear Flash or by downloading new firmware. You *must* document the password because it is not possible for you to restore a unit to the default, no-passwords condition.

Chapter 4. Configuration

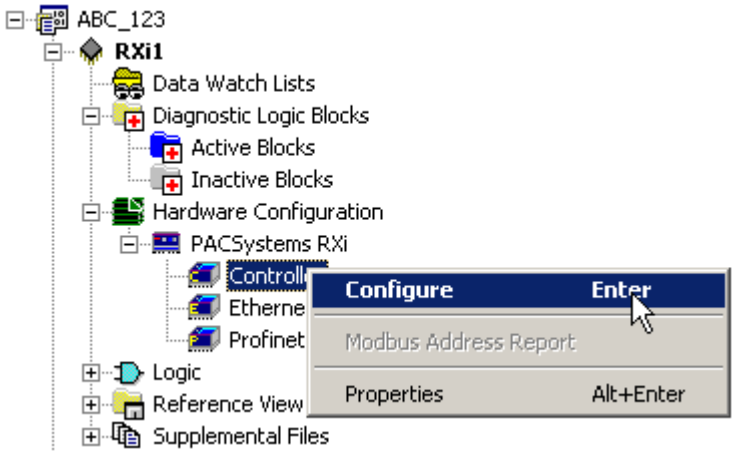
The PACSystems RXi Controller and PROFINET I/O system is configured using Proficy Machine Edition Logic Developer-PLC programming software. Refer to the *Machine Edition Logic Developer-PLC Getting Started Manual*, GFK-1918 and the online help for a description of configuration functions.

To configure the PROFINET interface, refer to 4.2, “Configuring the Embedded PROFINET Controller.” For details on configuring the Ethernet interface, refer to 4.3, "Configuring the Embedded Ethernet Interface.”

4.1 Configuring Controller Operation

To configure the RXi Controller’s embedded CPU functionality using the PME Logic Developer-PLC programming software:

1. In the Project tab of the Navigator, expand your PACSystems RXi target and the hardware configuration.
2. Right click the Controller and choose Configure. The Parameter Editor window displays the CPU parameters.



3. To edit a parameter value, select the desired tab, then click in the appropriate Values field.
4. Store the configuration to the Controller so these settings can take effect. For details, see “4.4, Storing (Downloading) Hardware Configuration.”

These parameters specify basic operating characteristics of the CPU. For additional details on how these parameters affect CPU operation, refer to Chapter 5.

4.1.1 Controller Settings Parameters

Settings Parameters	
Passwords	Specifies whether passwords are Enabled or Disabled. Default: Enabled. **Notes:** If Enhanced Security is enabled in the target properties, the Passwords setting will be Enabled and read-only, and the Access Control tab (page 46) appears. When passwords are disabled, they cannot be re-enabled without clearing Controller memory.
Stop-Mode I/O Scanning	Specifies whether the I/O is scanned while the Controller is in Stop mode. Default: Disabled
Watchdog Timer (ms)	(Milliseconds in 10 ms increments.) Requires a value that is greater than the program sweep time. The software watchdog timer is designed to detect "failure to complete sweep" conditions. The CPU restarts the watchdog timer at the beginning of each sweep. The watchdog timer accumulates time during the sweep. The software watchdog timer is useful in detecting abnormal operation of the application program, which could prevent the CPU sweep from completing within the watchdog time period. Valid range: 10 through 2550, in increments of 10. Default: 200.
Logic/Configuration Power-up Source	Specifies the location/source of the logic and configuration data that is to be used (or loaded/copied into RAM) after each power up. Choices: Always RAM, Always Flash, Conditional Flash. Default: Always RAM.
Data Power-up Source	Specifies the location/source of the reference data that is to be used (or loaded/copied into RAM) after each power up. Choices: Always RAM, Always Flash, Conditional Flash. Default: Always RAM.
Power-up Mode	Selects the CPU mode to be in effect immediately after power-up. Choices: Last, Stop, Run. Default: Last (the mode it was in when it last powered down). **Notes:** If Logic/Configuration Power-up Source is set to Always RAM, the CPU powers up in Stop mode regardless of the setting of the Power-up Mode parameter. If Power-up Mode is set to Last and Logic/Configuration Power-up Source is set to Always Flash or Conditional Flash, the CPU powers up in Run/Disabled mode.
Modbus Address Space Mapping Type	Specifies the type of memory mapping to be used for data transfer between Modbus TCP/IP clients and the PACSystems controller. Choices: **Disabled:** The “Disabled” setting is intended for use in systems containing Ethernet firmware that does not support Modbus TCP. **Standard Modbus Addressing:** Causes the Ethernet firmware to use the standard map, which is displayed on the Modbus TCP Address Map tab. Default: Disabled For details on the PACSystems RXi implementation of Modbus TCP server, refer to”9.6, Modbus TCP Client (Channels) Operation.”

4.1.2 Controller Scan Parameters

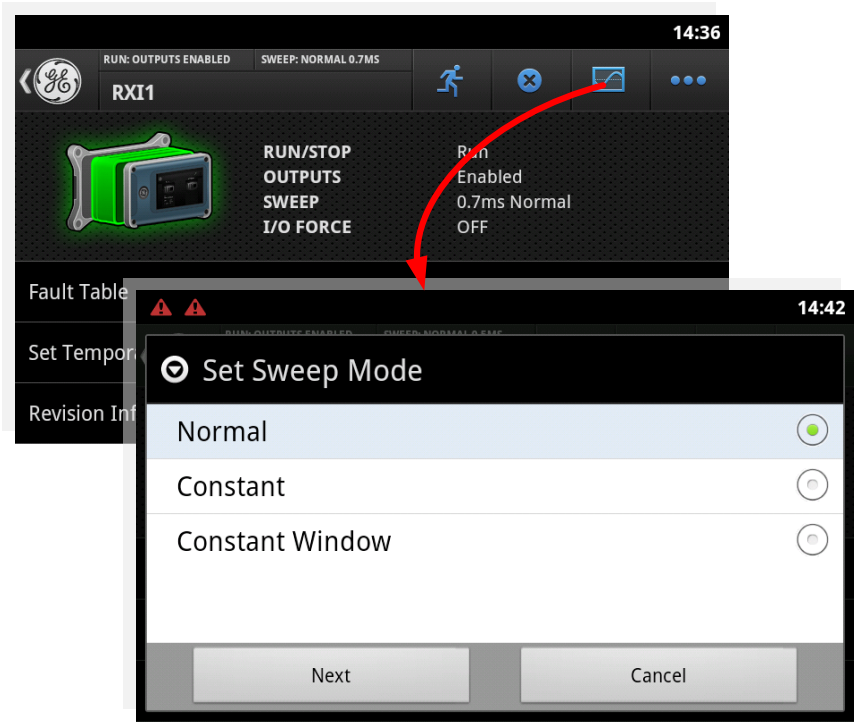
These parameters determine the characteristics of CPU sweep execution.

Scan Parameters	
Sweep Mode	The sweep mode determines the priority of tasks the CPU performs during the sweep and defines how much time is allotted to each task. The parameters that can be modified vary depending on the selection for sweep mode. The Controller Communications Window, Backplane Communications Window, and Background Window phases of the sweep can be run in various modes, based on the sweep mode. Choices: ■ Normal mode: The sweep executes as quickly as possible. The overall Controller sweep time depends on the logic program and the requests being processed in the windows and is equal to the time required to execute the logic in the program plus the respective window timer values. The window terminates when it has no more tasks to complete. This is the default value. ■ Constant Window mode: Each window operates in a Run-to-Completion mode. The Controller alternates among three windows for a time equal to the value set for the window timer parameter. The overall sweep time is equal to the time required to execute the logic program plus the value of the window timer. This time may vary due to sweep-to-sweep differences in the execution of the program logic. ■ Constant Sweep mode: The overall sweep time is fixed. Some or all of the windows at the end of the sweep might not be executed. The windows terminate when the overall sweep time has reached the value specified for the Sweep Timer parameter.
Logic Checksum Words	The number of user logic words to use as input to the checksum algorithm each sweep. Valid range: 0 through 32760, in increments of 8. Default: 16.
Controller Communication Window Mode	(Available only when Sweep Mode is set to *Normal.)* Execution settings for the Controller Communications Window. Choices: ■ Complete: The window runs to completion. There is no time limit. ■ Limited: Time sliced. The maximum execution time for the Controller Communications Window per scan is specified in the Controller Communications Window Timer parameter. Default: Limited.
Controller Communications Window Timer (ms)	(Available only when Sweep Mode is set to *Normal.* Read-only if the Controller Communications Window Mode is set to Complete.) The maximum execution time for the Controller Communications Window per scan. This value cannot be greater than the value for the watchdog timer. The valid range and default value depend on the Controller Communications Window Mode: ■ Complete: There is no time limit. ■ Limited: Valid range: 0 through 255 ms. Default: 10. **Caution** **Setting the Limited Controller Communications Window Timer too low will constrain the ability of the IDM to communicate with the controller. This will interfere with the ability of the IDM to report controller status and to command controller state changes.**

Scan Parameters	
Backplane Communication Window Mode	(Available only when Sweep Mode is set to *Normal.*) Execution settings for the Backplane Communications Window. Choices: ■ Complete: The window runs to completion. There is no time limit. ■ Limited: Time sliced. The maximum execution time for the Backplane Communications Window per scan is specified in the Backplane Communications Window Timer parameter. Default: Complete. ***Note:*** Even though the RXi Controller does not operate on a backplane, the Backplane Communications window still runs and settings that affect it will be fulfilled.
Backplane Communications Window Timer (ms)	(Available only when Sweep Mode is set to *Normal.* Read-only if the Backplane Communications Window Mode is set to *Complete.*) The maximum execution time for the Backplane Communications Window per scan. This value can be greater than the value for the watchdog timer. The valid range and the default depend on the Backplane Communications Window Mode: Complete: There is no time limit. The Backplane Communications Window Timer parameter is read-only. Limited: Valid range: 0 through 255 ms. Default: 10ms
Background Window Timer (ms)	(Available only when Sweep Mode is set to *Normal.*) The maximum execution time for the Background Communications Window per scan. This value cannot be greater than the value for the watchdog timer. Valid range: 0 through 255 Default: 0
Sweep Timer (ms)	(Available only when Sweep Mode is set to *Constant Sweep.*) The maximum overall scan time. This value cannot be greater than the value for the watchdog timer. Some or all of the windows at the end of the sweep might not be executed. The windows terminate when the overall Controller sweep time has reached the value specified for the Sweep Timer parameter. Valid range: 5 through 2550, in increments of 5. If the value typed is not a multiple of 5ms, it is rounded to the next highest valid value. Default: 100.
Window Timer (ms)	(Available only when Sweep Mode is set to *Constant Window.*) The maximum combined execution time per scan for the Controller Communications, Backplane Communications, and Background Communications windows. This value cannot be greater than the value for the watchdog timer. Valid range: 3 through 255, in increments of 1. Default: 10.
Number of Last Scans	The number of scans to execute after the CPU receives an indication that a transition from Run to Stop mode should occur. (Used for Stop and Stop Fault, but not Stop Halt.) Choices: 0, 1, 2, 3, 4, 5. Defaults: ▪ 0 when creating a new PACSystems RXi target or converting a Series 90-70 target to RXi. ▪ 1 when converting a Series 90-30 target to RXi.

4.1.2.1 Using the IDM to select Sweep modes and Communications Window Values

You can use the IDM to change the CPU sweep mode and values for the communications windows as follows:



To access additional screens, touch the Next button.

4.1.3 Controller Memory Parameters

The PACSystems user memory contains the application program, hardware configuration (HWC), registers (%R), bulk memory (%W), analog inputs (%AI), analog outputs (%AQ), and managed memory.

Managed memory consists of allocations for symbolic variables and I/O variables. The symbolic variables feature allows you to create variables without having to manually locate them in memory. An I/O variable is a symbolic variable that is mapped to a module's inputs and outputs in the hardware configuration. For details on using symbolic variables and I/O variables, refer to 7.1 Variables.

The amount of memory allocated to the application program and hardware configuration is automatically determined by the actual program (including logic, C data, and %L and %P), hardware configuration, and symbolic variables created in the programming software. The rest of the user memory can be configured to suit the application. For example, an application may have a relatively large program that uses only a small amount of register and analog memory. Similarly, there might be a small logic program but a larger amount of memory needed for registers and analog inputs and outputs.

Appendix C provides a summary of items that count against user memory.

4.1.3.1 Calculation of Memory Required for Managed Memory

The total number of bytes required for symbolic and I/O variables is calculated as follows:

[((number of symbolic discrete bits) × 3) / (8 bits/byte)]
\+ [((number of I/O discrete bits) × Md) / (8 bits/byte)]
\+ [(number of symbolic words × (2 bytes/word)]
\+ [(number of I/O words) × (Mw bytes/word)]

Md = 3 or 4. The number of bits is multiplied by 3 to keep track of the force, transition, and value of each bit. If point faults are enabled, the number of I/O discrete bits is multiplied by 4.

Mw = 2 or 3. There are two 8-bit bytes per 16-bit word. If point faults are enabled, the number of bytes is multiplied by 3 because each I/O word requires an extra byte.

4.1.3.2 Calculation of Total User Memory Configured

The total amount of configurable user memory (in bytes) configured in the CPU is calculated as follows:

Total managed memory (bytes)
\+ Total reference words × (2 bytes/word)
\+ *[if Point Faults are enabled]* (Total words of %AI memory + total words of %AQ memory) × (1 byte / word)
\+ *[if Point Faults are enabled]* (Total bits of %I memory + total bits of %Q memory) / 8 bits/byte)

Note: The total number of reference points is considered system memory and is not counted against user memory.

Memory Allocation Configuration

Memory Parameters	
Reference Points	
%I Discrete Input, %Q Discrete Output, %M Internal Discrete, %S System, %SA System, %SB System, %SC System, %T Temporary Status, %G Genius Global	The upper range for each of these memory types. Read only.
Total Reference Points	Read only. Calculated by the programming software.
Reference Words	
%AI Analog Input	Valid range: 0 through 32,640 words. Default: 64
%AQ Analog Output	Valid range: 0 through 32,640 words. Default: 64
%R Register Memory	Valid range: 0 through 32,640 words. Default: 1024.
%W Bulk Memory	Valid range: 0 through maximum available user RAM. Increments of 2048 words. Default: 0.
Total Reference Words	Read only. Calculated by the programming software.
Managed Memory	
Symbolic Discrete (Bits)	The configured number of bits reserved for symbolic discrete variables. Valid range: 0 through 83,886,080 in increments of 32768 bits. Default: 32,768.
Symbolic Non-Discrete (Words)	The configured number of 16-bit register memory locations reserved for symbolic non-discrete variables. Valid range: 0 through 5,242,880 in increments of 2048 words. Default: 65,536.
I/O Discrete (Bits)	The configured number of bits reserved for discrete IO variables. Valid range: 0 through 83,886,080 in increments of 32768 bits. Default: 0
I/O Non-Discrete (Words)	The configured number of 16-bit register memory locations reserved for non-discrete IO variables. Valid range: 0 through 5,242,880 in increments of 2048 words. Default: 0
Total Managed Memory Required (Bytes)	Read only. See page 42 for calculation.
Total User Memory Required (Bytes)	Read only. See page 42 for calculation.
Point Fault References	The Point Fault References parameter must be enabled if you want to use fault contacts in your logic. Assigning point fault references causes the CPU to reserve additional memory. When you download both the HWC and the logic to the Controller, the download routine checks if there are fault contacts in the logic and if there are, it checks if the HWC has the Point Fault References parameter set to Enabled. If the parameter is Disabled, an error is displayed in the Feedback Zone. When you download only logic to the Controller, the download routine checks if there are fault contacts in the logic and if there are, it checks if the HWC on the Controller has the Point Fault References parameter set to Enabled. If the parameter is Disabled, an error is displayed in the Feedback Zone. Default: Disabled

4.1.4 Fault Parameters

You can configure each fault action to be either diagnostic or fatal.

A ***diagnostic fault*** does not stop the Controller from executing logic. It sets a diagnostic variable and is logged in a fault table.

A ***fatal fault*** transitions the Controller to the Stop Faulted mode. It also sets a diagnostic variable and is logged in a fault table.

Fault Parameters	
I/O Controller or I/O Bus Fault	(Fault group 9.) When a bus fault, a global memory fault, or an IOC hardware fault occurs, system variable #IOC_FLT (%SA22) turns on. (To turn it off, cycle power or clear the I/O Fault table.) Default: Diagnostic
CPU Over Temperature	(Fault group 24, error code 1.) When the operating temperature of the CPU exceeds the normal operating temperature, system variable #OVR_TMP (%SA8) turns ON. (To turn it OFF, clear the controller fault table or reset the Controller.) Default: Diagnostic.
Controller Fault Table Size	(Read-only.) The maximum number of entries in the Controller Fault Table. Value set to 64.
I/O Fault Table Size	(Read-only.) The maximum number of entries in the I/O Fault Table. Value set to 64.

4.1.5 Scan Sets Parameters

You can create multiple sets of asynchronous I/O scans, with a unique scan rate assigned to each scan set. You can assign up to 31 scan sets for a total of 32. Scan set 1 is the standard scan set where I/O is scanned once per sweep. Each module is assigned to a scan set in the module's configuration. Scan Set 1 is the default scan set.

Scan Set Parameters	
Number	A sequential number from 1 to 32 is automatically assigned to each scan set. Scan set 1 is reserved for the standard scan set.
Scan Type	Determines whether the scan set is enabled (as a fixed scan) or is disabled. Choices: Disabled, Fixed Scan. Default: Disabled.
Number of Sweeps	(Editable only when the Scan Type is set to Fixed Scan.) The scan rate of the scan set. A value of 0 prevents the I/O from being scanned. Valid range: 0 through 64. Default: 1.
Output Delay	(Editable only when the Number of Sweeps is non-zero.) The number of sweeps that the output scan is delayed after the input scan has occurred. Valid range: 0 to (number of Sweeps - 1) Default: 0.
Description	(Editable only when the Scan Type is set to Fixed Scan.) Brief description of the scan set (32 characters maximum).

4.1.6 Modbus TCP Address Map

This read-only tab displays the standard mapping assignments between Modbus address space and the CPU address space. Ethernet interfaces in the PACSystems controller use Modbus-to-Controller address mapping based on this map.

Modbus Register	The Modbus protocol uses five reference table designations: 0xxxx Coil Table. Mapped to the %Q table in the CPU. 1xxxx Input Discrete Table. Mapped to the %I table in the CPU. 3xxxx Input Register Table. Mapped to the %AI register table in the CPU. 4xxxx Holding Register Table. Mapped to the %R table in the CPU. 6xxxx File Access Table. Mapped to the %W table in the CPU.
Start Address	Lists the beginning address of the mapped region.
End Address	Lists the ending address of the mapped region. For word memory types (%AI, %R and %W) the highest address available is configured on the Memory tab.
Controller Memory Address	Lists the memory type of the mapped region.
Length	Displays the length of the mapped region.

4.1.7 Access Control

The Access Control list allows you to specify the reference address ranges that can be accessed by non-local devices such as HMIs and other controllers. To use this feature, Enhanced Security must be enabled in the target's properties.

When Enhanced Security mode is enabled, any reference address range not defined can ***not*** be accessed by other devices. External reads and writes that do not exist in the table are rejected by the firmware.

If overlapping memory ranges are defined, they must have the same Access level.

For symbolic variables, access control is specified by the variable's *Publish* property, which includes a Read Only and Read/Write setting.

Note: When requesting data from an external device, some drivers packetize data to optimize communication. If a request attempts to read a value that is not published, the entire packet will fail. A fault has been added to the fault table to help you understand a failed read/write. After addressing the fault, you must clear the fault in order to try again.

When you go online with the controller with enhanced security enabled, you will be taken to Privilege Level 1 and PME displays a security icon.

Memory Area	The memory area in which the reference address range is defined. Default: Select an Area Choices: %AI Analog Input, %AQ Analog Output, %I Discrete Input, %G Genius Global, %M Internal Discrete, %Q Discrete Output, %R Register Memory, %S System, %SA System, %SB System, %SC System, %T Temporary Status, %W Bulk Memory.
Start	The starting offset of the reference address range. Default: 0 (not valid) Valid range: For %S, %SA, %SB and %SC, must be 1. All other memory types: 1 through the upper limit of the reference address range. Must be less than the End value.
End	The ending offset of the reference address range. Default: 0 (not valid) Valid range: For %S, %SA, %SB and %SC, must be 128. All other memory types: Any value greater than Start, through the upper limit of the reference address range. For word memory types (%AI, %R and %W) the highest address available is configured on the Memory tab.
Access	Selects the type of external access allowed for the defined address range. Choices: Read-Only, Read/Write Default: Read-Only

4.2 Configuring the Embedded PROFINET Controller (PNC)

This section explains how to configure the RXi PNC and its IO-devices in a PACSystems RXi Controller system.

The Proficy Machine Edition programming software is used to create and download the configuration for an RXi PROFINET network and its devices. Additional information about RXi configuration is available in the Proficy Machine Edition online help.

This section discusses the following topics:

- System Planning
- Basic Configuration Steps
- Configuring the PNC
  - The PNC's LAN
  - Configuring PNC Parameters
- Configuring PROFINET LANs
  - Configuring the LAN Properties
- Adding GE Intelligent Platforms IO-Devices to a LAN
- Adding Third-Party IO-Devices to a LAN
  - Editing Third-Party IO-Device Parameters
  - Configuring Sub-modules of an IO-Device
- Viewing / Editing IO-Device Properties
- Assigning IO-Device Names
- Configuring IO-Devices
- After the Configuration is Stored to the RXi CPU
  - Clearing the RXi Controller Configuration

4.2.1 System Planning for PROFINET Networks

4.2.1.1 Device Ownership

Note that all modules in an IO-Device must belong to the same PROFINET network. That is, they are all owned by the same PNC. Modules on an IO-Device cannot be split between PNCs.

4.2.1.2 PNC Loading Limits

To prevent overloading of the PNC, the maximum number of IO-Devices that can be configured is limited to the equivalent of eight devices with update rates of 1ms.

Devices configured with a longer update periods present smaller data loads to the PNC. For example, a device with an update rate of 2ms is equivalent to ½ device at 1ms.

The maximum number of devices allowed (up to 128) is determined by their update rates as shown in the following calculation:

$$SUM \frac{\text{Number of Devices at Update Rate } x}{\text{Update Rate } x} \leq 8 \text{ devices/ms}$$

4.2.1.3 Maximum Configuration at Each Update Rate

Update Rate	Total Number of Devices per PNC
1	8
2	16
4	32
8	64
16 – 512	128 (total device limit)

If the configuration exceeds the equivalent of eight devices with 1ms update rates, PROFICY Machine Edition will not store the configuration.

4.2.2 Basic Configuration Steps

The basic configuration steps for a PROFINET network are:

- Configure the parameters of the PNC.
- Select the PNC and add IO-Devices to its LAN. These IO-Devices can be GE Intelligent Platforms PNS modules or third-party IO-Devices. PNS modules and third-party IO-Devices use GSDML files to describe their capabilities. PME imports these GSDML files and incorporates the devices into the configuration.

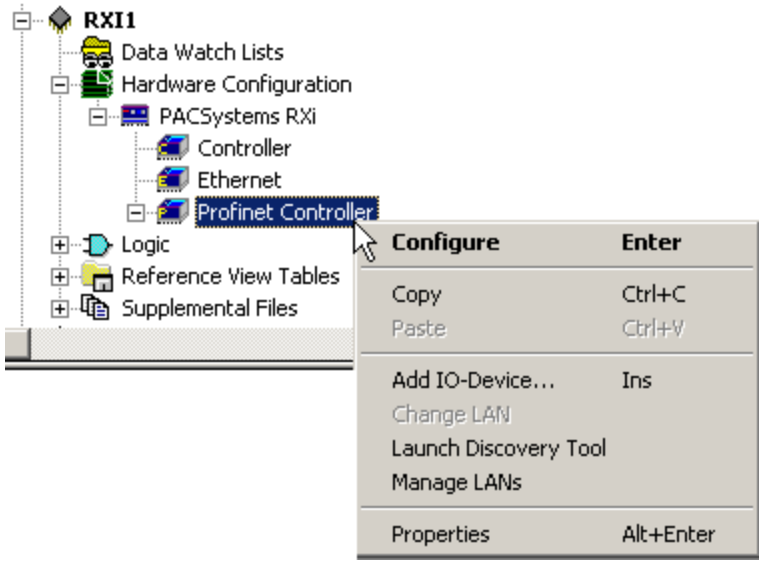
Configure the parameters of the IO-Devices.

- Configure the communications properties of the PNC, PNS modules, and third-party IO-Devices in the PME Inspector pane.
- Add modules to the IO-Device remote nodes.
- Configure the parameters of the modules and sub-modules in the remote nodes.
- When the configuration is ready, use the Discovery and Configuration Protocol (DCP) tool in Machine Edition to assign a name to each IO-Device so the PNC can connect to the devices and deliver their configuration.
- Store the configuration data from the programmer to the RXi Controller.

4.2.3 Configuring the Embedded PNC

This section describes one technique to configure the parameters of the RXi embedded PNC. The PME InfoViewer describes alternate menus and keyboard operations that can be used to perform the same functions.

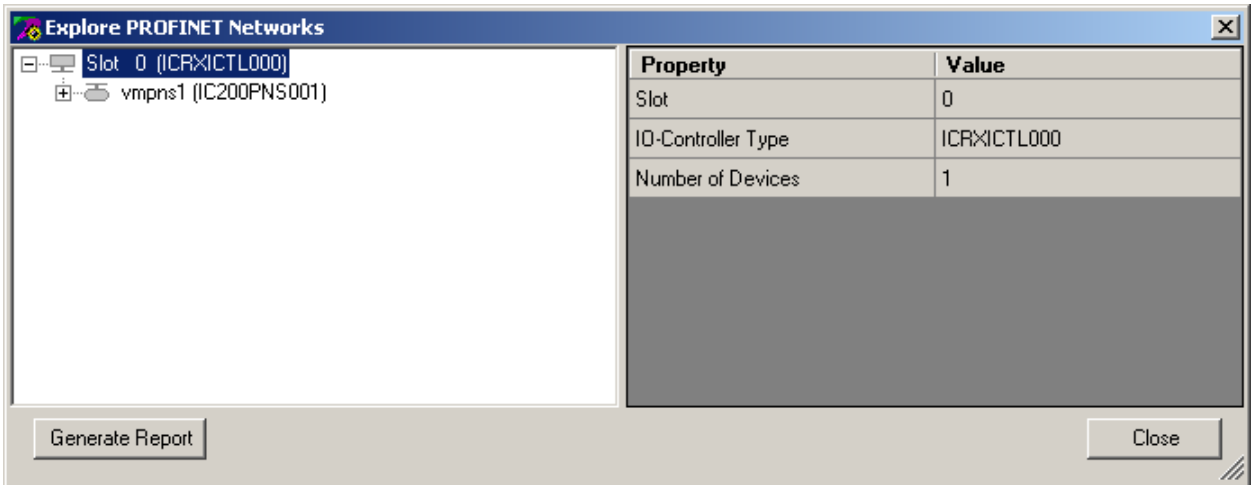
1. In the Project tab of the Navigator, expand the PACSystems RXi Target.
2. Click the PROFINET Controller. The PNC parameters are displayed in the Parameter Editor window. Its communications properties appear in the Inspector pane.



3. Edit the PNC's parameters and its communications properties as described in this chapter.

4.2.3.1 Exploring PROFINET Networks

To explore the PROFINET networks in the system while PROFICY Machine Edition is online with the RXi system, right-click on the Target icon (not the PNC) and select Explore PROFINET Networks under the Online Commands menu item. The PNC will show version information about all the devices on its network such as PNS modules and modules in remote nodes, whether the devices are configured or not.

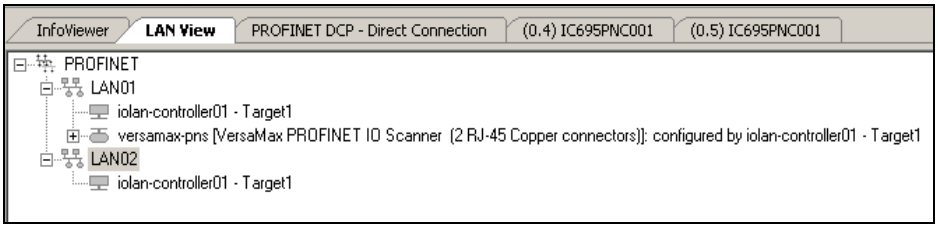


4.2.3.2 Viewing the PROFINET Controller's LAN

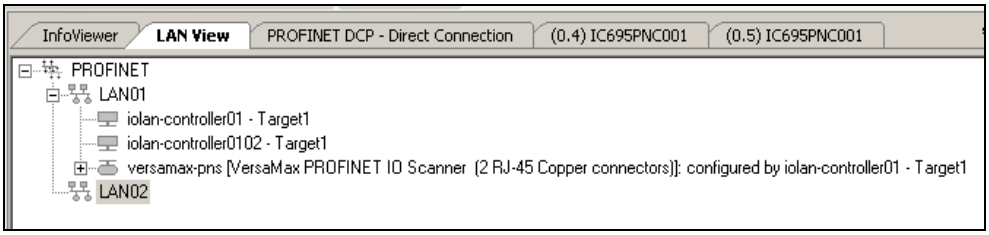
To view the LANs in the project, click Tools in the PROFICY Machine Edition toolbar, and select LAN View, or right-click the PNC and select Manage LANs.

Adding the first RXi target to the project automatically creates a LAN for it. For each subsequent RXi target, an existing LAN can be selected or a new LAN can be created. Opening the LAN View shows PNCs on their assigned LANs.

A LAN can also be added to the project by right-clicking the PROFINET icon in the LAN View and selecting Add LAN.



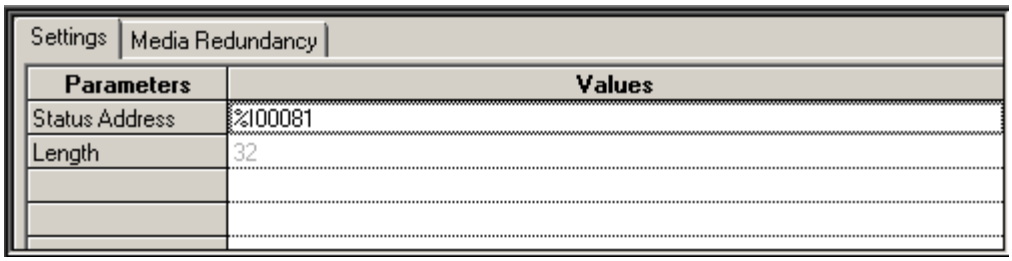
A PNC can be moved to a different LAN by selecting the module in the Navigator and dragging it to the target LAN. Here, the PNC has been moved from LAN02 (see above) to LAN01:



4.2.3.3 Configuring PROFINET Controller Parameters

Configure the PNC parameters by editing the tabs as appropriate. Additional settings can be viewed and edited in the module properties, which are displayed in the Inspector window.

Settings



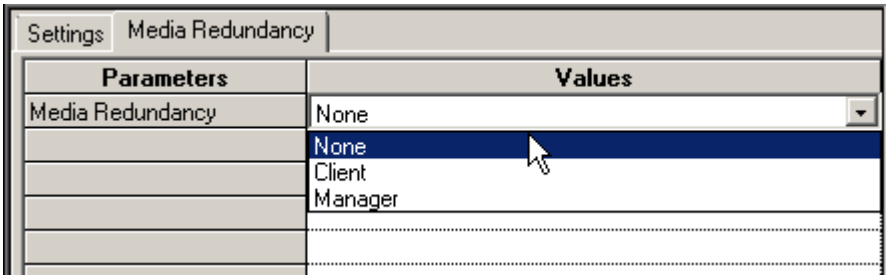
Status Address, Length: The Status Address is the reference memory location for the PNC's 32 bits of status data. The Status address can be assigned to valid %I, %Q, %R, %AI, %AQ, %W, %G, %T or %M memory by right-clicking on the Status Address field and selecting the Data Entry tool. The default value is the next available %I address. See 11.8, PROFINET Controller Diagnostics for definitions of the status bits that the module writes to this address.

If Variable Mode is set to True in the PROFINET Controller properties, the Terminals Tab will be displayed instead of the Settings.

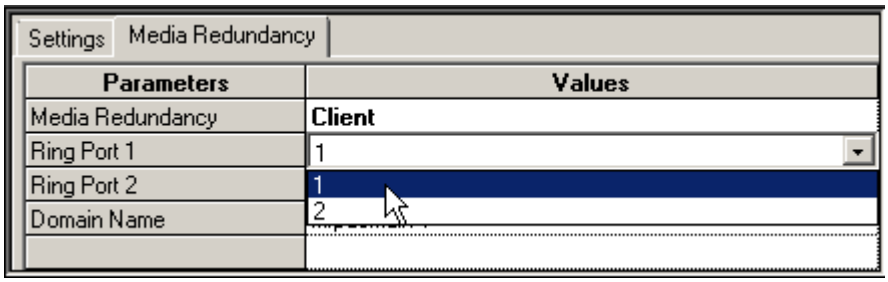
Note: Because point faults are not supported with %G, %T and %M, the other memory types or I/O Symbolics are preferred.

Media Redundancy Tab

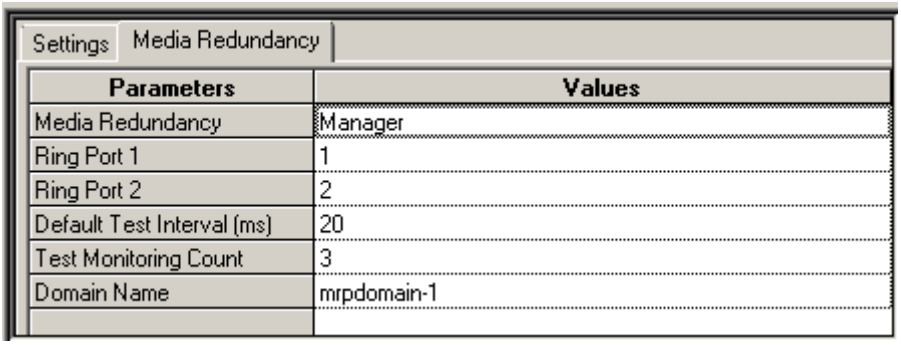
By default, a PNC is not set up for Media Redundancy. If the system will use Media Redundancy (see Chapter 12, PROFINET Redundant Media for more information), open the Media Redundancy Tab and select either Client or Manager:



If the PNC will be a Media Redundancy Client, click on Ring Port 1 and Ring Port 2 to choose the module ports then select the Domain Name that will be used.



If the PNC will be a Media Redundancy Manager, edit the Ring Port settings as above. You can also change the Default Test Interval in the range of 10 to 1000ms and the Test Monitoring Count (2 to 10). For the Media Redundancy Manager, the Domain Name can be edited by typing over the default name.



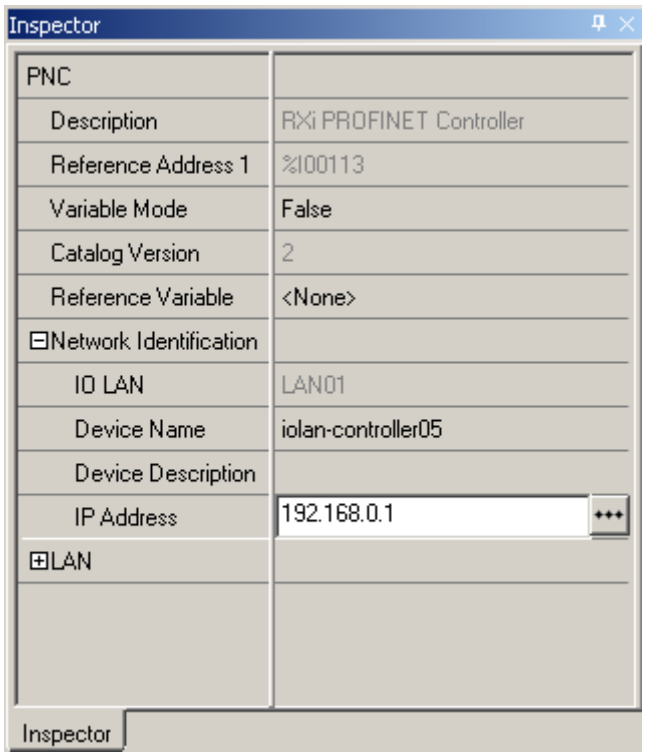
Note: In an MRP ring with a large number of clients, storing a configuration that causes all clients to reconfigure (for example, changing the Domain Name) may generate a large number of Loss/Addition of Device faults. This is expected behavior and all devices should automatically return to operational.

Terminals Tab

This configuration tab is displayed only when the PROFINET Controller's Variable Mode property is set to True. When Variable Mode is selected, the Status bits are referenced as I/O variables that are mapped to the status bits on this configuration tab.

The use of I/O variables allows you to configure the PNC without having to specify the reference addresses to use for the status information. Instead, you can directly associate variable names with the status bits. For more information, refer to "7.1.3, I/O Variables."

4.2.3.4 Configuring PROFINET Controller Properties



Description: (Read-only) Description of the controller type.

Reference Address 1: Starting address for status bits assigned in the Settings tab. (Not displayed if Variable Mode is set to True.)

Variable Mode: When enabled (set to True), the Status bits are referenced as I/O variables that are mapped to the status bits on the Terminals tab.

Reference Variable: (Optional) Allows you to create a reference ID variable to identify the PNC. For details, see "11.7.1, Reference ID Variables for the RXi Application."

Network Identification: (Read only) The name of the IO LAN that the PNC belongs to.

Device Name: The name that identifies the PNC on the LAN. Must be unique on the IO LAN.

Device Description: (Optional) A user-defined description of the PNC.

IP Address: The IP address of the PNC. Must be not be used for any other device in the LAN. PME sets the default value to the lowest available IP address in the valid range.

LAN: These properties apply to the PROFINET LAN. You can view them in the Inspector when you select a LAN controller in the hardware configuration, or in LAN View.

4.2.3.5 Non-Volatile Configuration Parameters and Properties for PNC

These values are updated the first time a hardware configuration is stored to the RXi. They are ***not*** reset to the default values when configuration is cleared; they can only change when a new hardware configuration is stored.

Parameter	*Description*	*Default Value*
IP Address	IP Address of the PNC.	0.0.0.0
Subnet Mask	Subnet mask of the PNC.	0.0.0.0
Default Gateway	Default Gateway for the PNC.	0.0.0.0
Device Name	PNC's PROFINET Device Name. Defaults to empty string indicating the PNC is not named.	" "
Media Redundancy	Media redundancy role: Specifies whether media redundancy is disabled (None), or if it is enabled as a Client or as a Manager.	None
Ring Port 1 ID	Indicates one of the two network ports involved in Media Redundancy. Valid range: 1 to 2.	1
Ring Port 2 ID	Indicates one of the two network ports involved in Media Redundancy. Valid range: 1 to 2.	2
Default Test Interval	Interval for sending test frames on ring ports in millisecond units. Valid range: 10 to 1000ms.	20 ms
Test Monitoring Count	Indicates the number of consecutive failed test frames before declaring a ring failure. Valid range: 2 to 10.	3

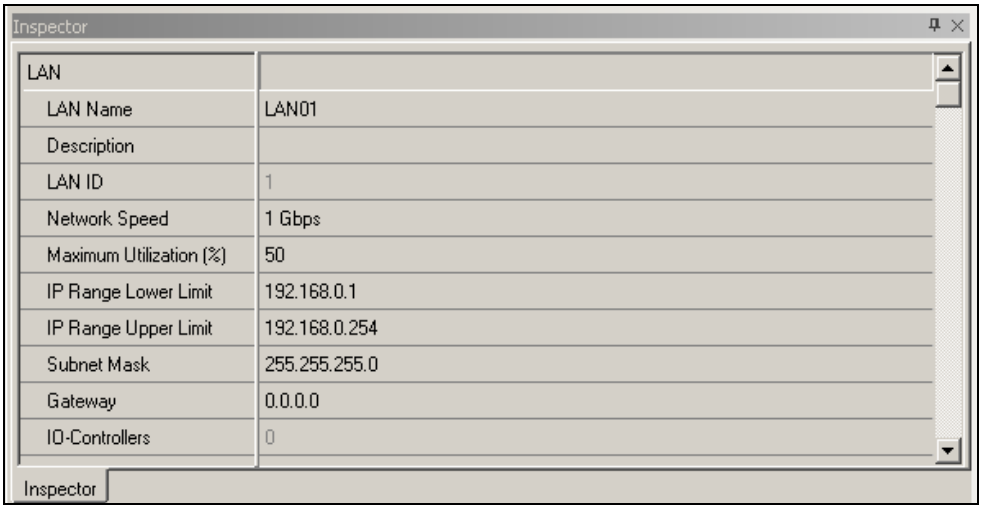
4.2.4 Configuring PROFINET LANs

To view the LANs in the project, click on Tools in the Machine Edition toolbar, and select LAN View from the menu.

Expand the LAN icon in the LAN View to see the devices it includes.

4.2.4.1 Configuring the LAN Properties

Machine Edition automatically assigns a set of default properties to the LAN. Select the LAN icon in the LAN viewer to display or edit its communications properties in the Inspector pane:



LAN Name: this can be edited or the default name can be used. Space characters are not permitted.

Description: an optional description of up to 255 characters can be entered for the LAN.

LAN ID: (read-only) this number identifies the LAN in the PROFICY Machine Edition project.

Network Speed: the bandwidth available on the network. The default of 1Gbps can be changed to 100Mbps.

Maximum Utilization (%): the maximum percentage of total network bandwidth that can be used for PROFINET I/O traffic. It can be edited to any value between 10 and 80 (do not enter the % character). Consider other network traffic when changing this parameter.

IP Range Lower Limit: the lowest IP address for automatically assigning IP addresses to PNCs and LAN devices. Default is 192.168.x.1, where x is the lowest number not used by another LAN in the project.

IP Range Upper Limit: the highest IP address for automatically assigning IP addresses to PNCs and LAN devices. Default is 192.168.x.254, where x is the lowest number not used by another LAN in the project.

Subnet Mask: Mechanism that filters network communications so that they are routed only to subnets to which they are addressed. The value defined here propagates to PNCs and I/O devices throughout the network.

Caution

If the subnet mask is improperly set, devices may be unable to communicate on the network and might disrupt network communications. Contact your network administrator to assign values that work with an existing network.

PROFINET can only communicate to nodes in the local subnet. All nodes on the LAN must be in the same subnet. The IP Range Lower/Upper Limits indicate the auto-assignment range. The range of the LAN is the subnet mask range. A node can be assigned to any address in the subnet (which may be outside of the auto-assignment range given in the dialog), but the addresses for all nodes must be in the subnet.

Gateway: The IP Address of the device that connects two (sub) networks that use different communications protocols, enabling them to communicate with each other. The value defined here propagates to PNCs and I/O devices throughout the network.

Caution

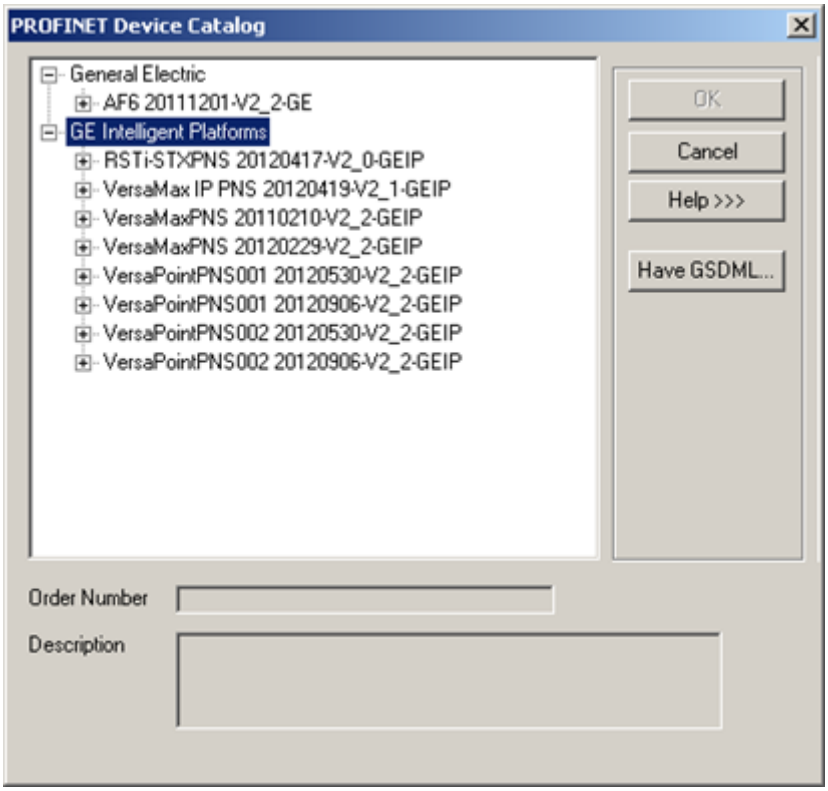
If the gateway is improperly set, devices may be unable to communicate on the network and might disrupt network communications. Contact your network administrator to assign values that work with an existing network.

IO-Controllers: (Read-only.) The number of I/O Controllers configured to reside on the LAN.

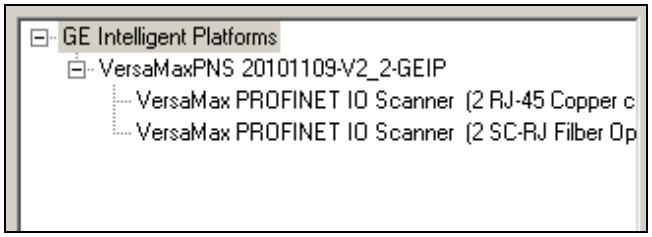
IO-Devices: (Read-only.) The number of I/O devices configured to reside on the LAN.

4.2.5 Adding a GE Intelligent Platforms PROFINET Scanner to a LAN

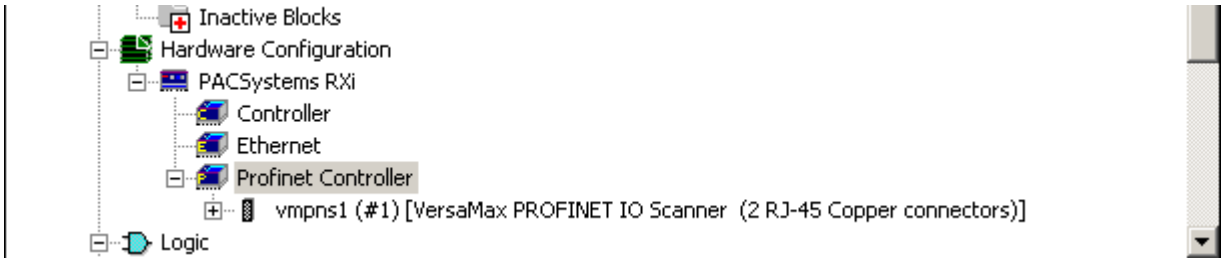
To add a PROFINET Scanner (PNS) to a LAN, in the Navigator, right-click the PNC and select Add IO-Device. The PROFINET Device Catalog, which is populated by the GSDML files installed with PME, appears.



In the PROFINET Device Catalog, expand the device family and choose the module type. The following examples show a VersaMax PNS.


GE Intelligent Platforms
VersaMaxPNS 20101109-V2_2-GEIP
VersaMax PROFINET IO Scanner (2 RJ-45 Copper c
VersaMax PROFINET IO Scanner (2 SC-RJ Filber Op

Select the PNS type and click OK. The PNS appears in the Navigator window:



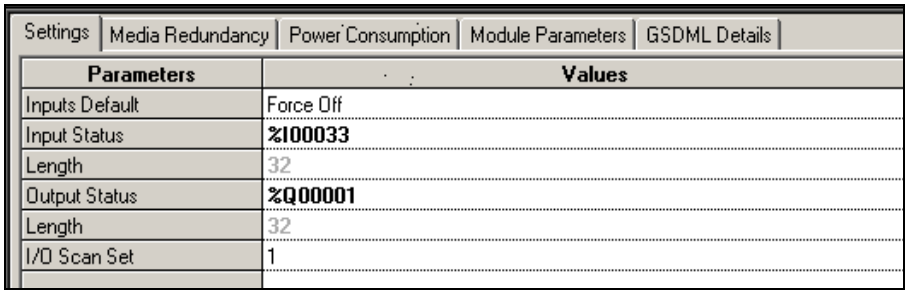
4.2.5.1 Configuring a PROFINET Scanner

After adding a PNS to the LAN, its parameters can be configured by either double-clicking the scanner in the Navigator, or right-clicking and selecting Configure from the menu.

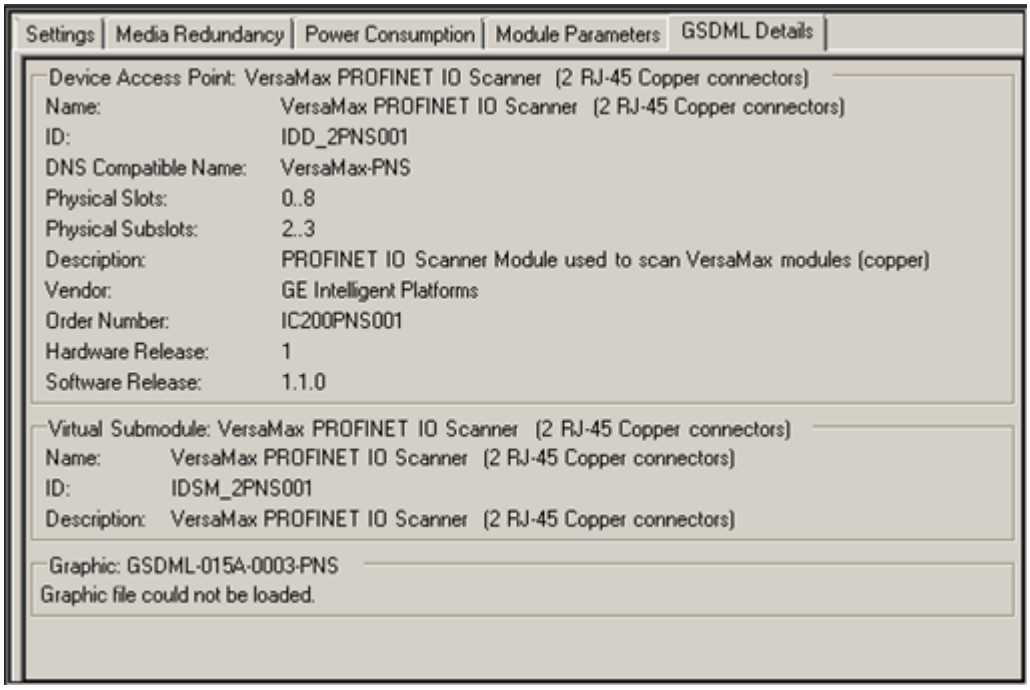
PROFINET Scanner Parameters

The parameters displayed will depend on the type of PNS. For configuration details, refer to the user manual for the PNS.

The following example shows the Settings for a VersaMax PNS.



The PNS module's GSDML tab displays the information from its GSDML file.


Settings
Media Redundancy
Power Consumption
Module Parameters
GSDML Details
Device Access Point: VersaMax PROFINET IO Scanner (2 RJ-45 Copper connectors)
Name: VersaMax PROFINET IO Scanner (2 RJ-45 Copper connectors)
ID: IDD_2PNS001
DNS Compatible Name: VersaMax-PNS
Physical Slots: 0..8
Physical Subslots: 2..3
Description: PROFINET IO Scanner Module used to scan VersaMax modules (copper)
Vendor: GE Intelligent Platforms
Order Number: IC200PNS001
Hardware Release: 1
Software Release: 1.1.0
Virtual Submodule: VersaMax PROFINET IO Scanner (2 RJ-45 Copper connectors)
Name: VersaMax PROFINET IO Scanner (2 RJ-45 Copper connectors)
ID: IDSM_2PNS001
Description: VersaMax PROFINET IO Scanner (2 RJ-45 Copper connectors)
Graphic: GSDML-015A-0003-PNS
Graphic file could not be loaded.

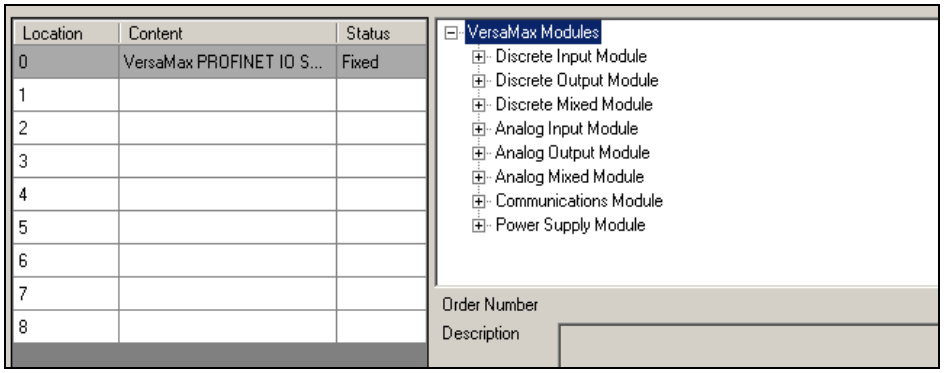
This information cannot be edited.

Note: For some PNS modules, additional GSDML information can be viewed by double-clicking components within the module's node in the Navigator view.

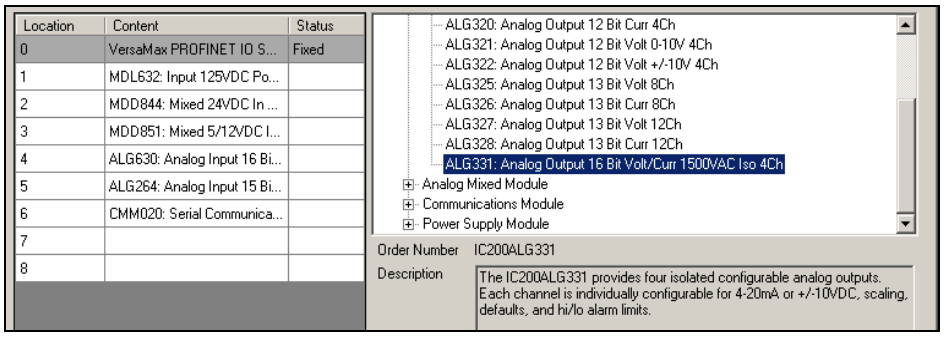
4.2.5.2 Adding Modules to a Remote Node

To add a module to the remote node, right click on the PNS icon in the Navigator and select Change Module List. In the right pane of the Change Module List window, expand the list of module types.

The following example shows the list of module types for a VersaMax PNS. For modules supported by a particular PNS type, refer to the documentation for that device.



Select modules from the list and drag them to their slot locations in the remote node.



(If you need to delete a module on the left, select it and press the keyboard Delete key). When the modules on the left are correct, click OK to add them to the configuration.

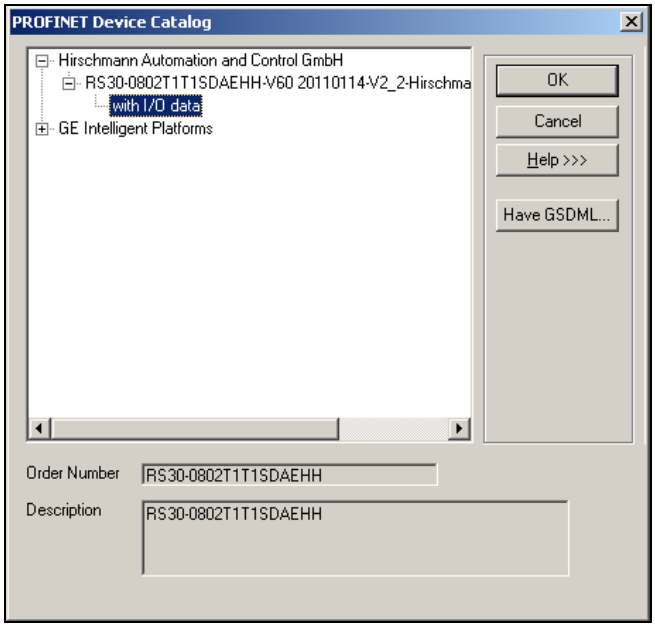
4.2.5.3 Configuring Module Parameters

After adding modules to the remote node, their parameters must be configured. This includes configuring a set of basic parameters (such as: reference address, length, scan set, carrier, report faults). For configuration details for those basic parameters refer to the documentation for the PNS module.

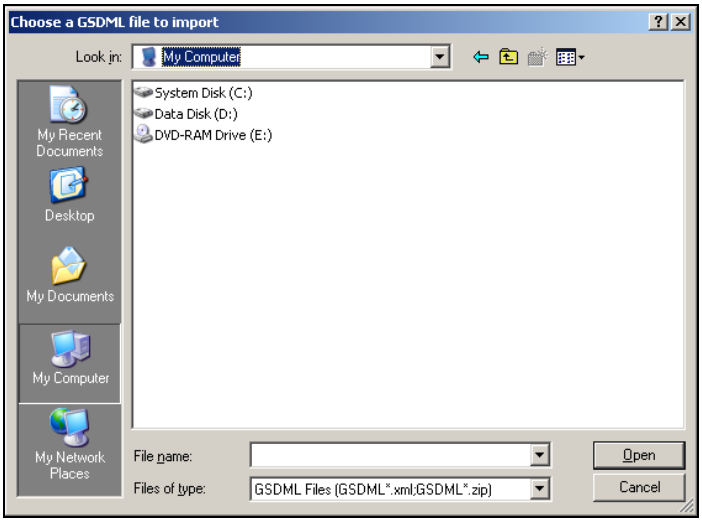
Some PNS modules support submodules. The configuration process for these is similar to that for modules.

4.2.6 Adding a Third-Party IO-Device to a LAN

To add a third-party IO-Device to a LAN, in the Navigator right-click on a PNC and select Add IO-Device. Choose the module type and click Have GSDML.



All third-party IO-Devices require GSDML files to be included in the system configuration. The GSDML file for the device must be present on the computer being used for the configuration.



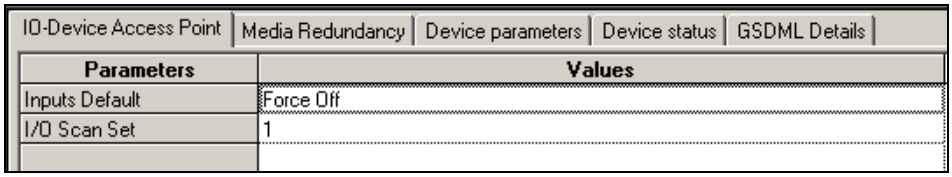
Provide a path to the file and click OK. The device is added to the configuration. PROFICY Machine Edition extracts necessary parameters from the GSDML file and makes the data available for editing within PROFICY Machine Edition.

4.2.6.1 Editing Third-Party IO-Device Parameters

To configure a third-party IO-Device's parameters, either double-click on the module in the Navigator, or right-click on the module and select Configure from the menu. Upon opening the IO-Devices configuration, you will see an IO-Device Access Point tab, a GSDML Details tab, and possibly additional parameter tabs (if defined by the device manufacturer in the associated GSDML file).

IO-Device Parameters (IO-Device Access Point Tab)

Use the IO-Device Access Point Tab to set up the device's interface to the RXi Controller:



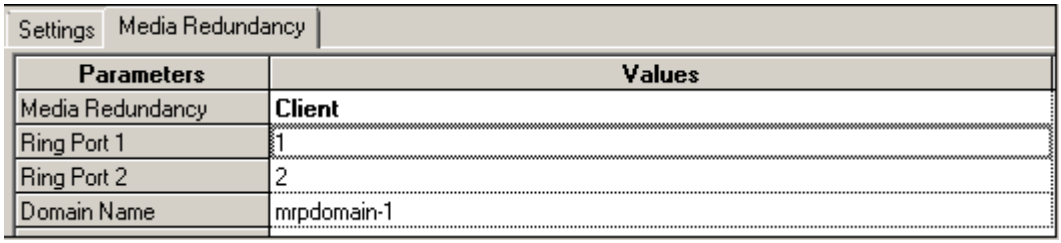
Inputs Default: Choose whether the RXi CPU will set inputs from the remote node to Off, or Hold Last State in the following cases:

- The PNC is not operational.
- The PNC cannot reach the device due to cable or network configuration issues.
- The device is not able to scan the sub-module in its remote node.

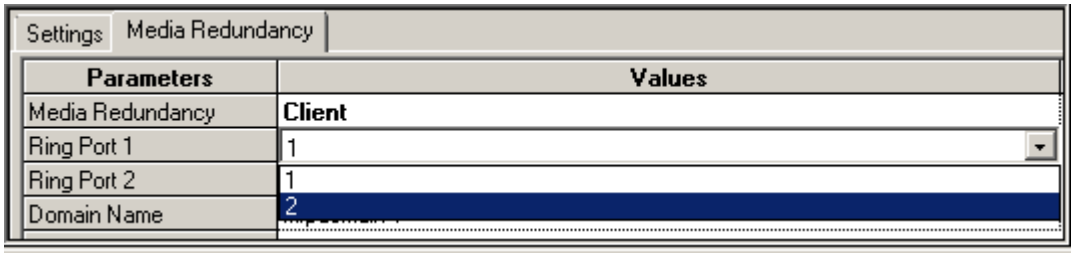
I/O Scan Set: The scan set for an IO-Device defaults to scan set 1. Scan sets are defined in the CPU's Scan Sets tab. The valid range is 1 through 32; the default value is 1.

IO-Device Parameters (Media Redundancy Tab)

If the IO-Device supports Media Redundancy (see Chapter 12, PROFINET Redundant Media for more information), a Media Redundancy Tab will be present. Open the Media Redundancy Tab and select either Client or Manager:



If the IO-Device will be a Media Redundancy Client, you can select which port will be used for Ring Port 1 and which will be used for Ring Port 2.



If the IO-Device will be a Media Redundancy Manager, you can edit the Ring Port selections as above. The device's GSDML file determines the valid ranges for Default Test Interval and Test Monitoring Count.

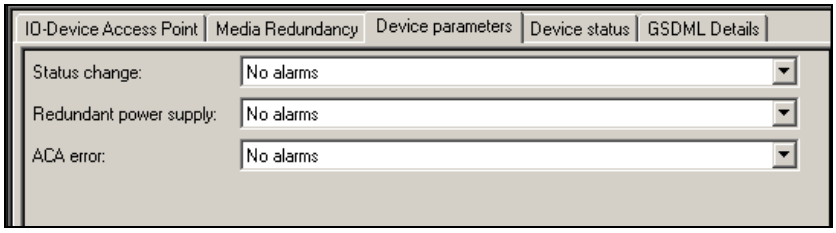
For the Media Redundancy Manager, the Domain Name can be edited by typing over the default name.

Parameters	Values
Media Redundancy	Manager
Ring Port 1	1
Ring Port 2	2
Default Test Interval (ms)	20
Test Monitoring Count	3
Domain Name	mrpdomain-1

IO-Device Parameters (Device Parameters Tab)

Additional Device Parameters tabs can be used to select additional device options, as defined by the device manufacturer.

Note: The names of the tabs and parameters are derived from manufacturer specific information contained in the associated GSDML file. For example:

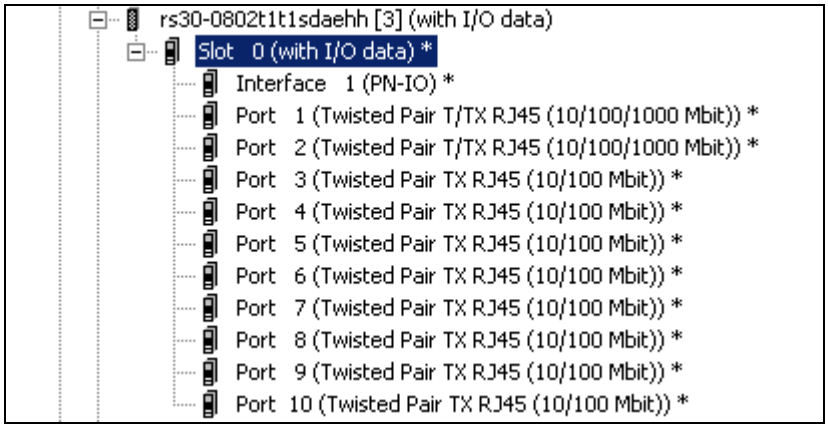


IO-Device Parameters (GSDML Details Tab)

The GSDML Details tab displays the device's GSDML parameters, which cannot be edited.

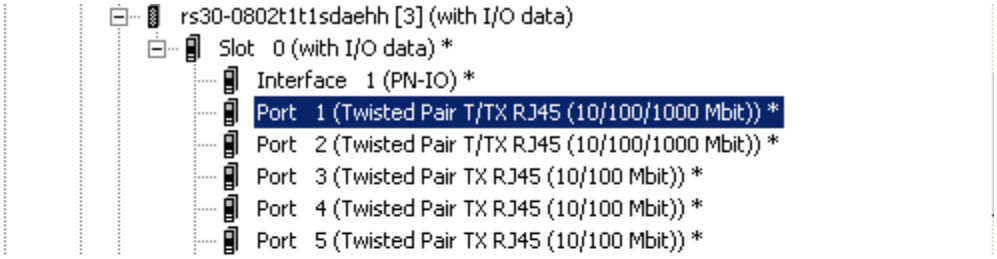
4.2.6.2 Configuring Sub-Modules of a Third-Party IO-Device

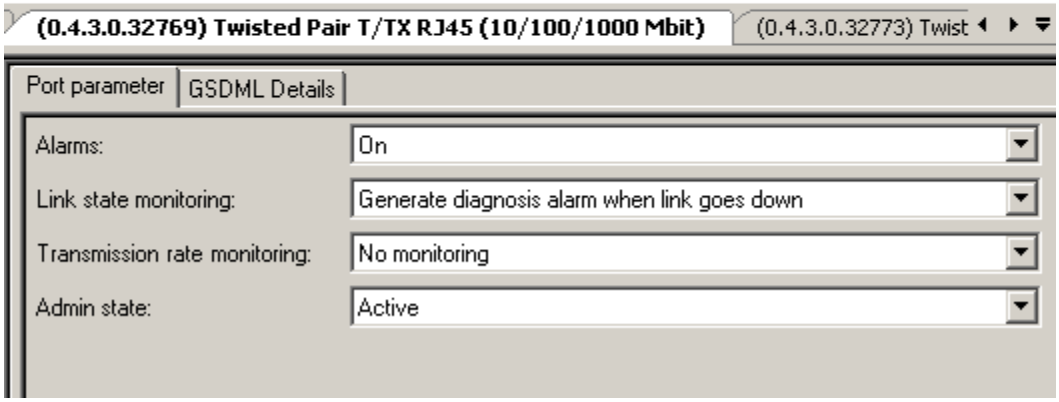
To configure the sub-modules of a third-party IO-Device, expand it in the Navigator. For example:


rs30-0802t1t1sdaehh [3] (with I/O data)
Slot 0 (with I/O data) *
Interface 1 (PN-IO) *
Port 1 (Twisted Pair T/TX RJ45 (10/100/1000 Mbit)) *
Port 2 (Twisted Pair T/TX RJ45 (10/100/1000 Mbit)) *
Port 3 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 4 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 5 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 6 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 7 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 8 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 9 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 10 (Twisted Pair TX RJ45 (10/100 Mbit)) *

This example's device is a switch. The parameters of its ports can be viewed and edited by either double-clicking on a port, or selecting a port then right-clicking and selecting Configure from the menu.

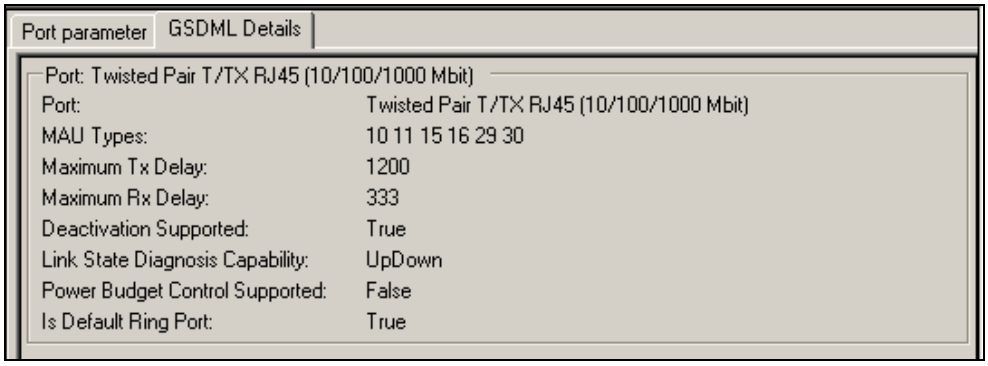
Select the individual ports, for the above example IO-Device, the port parameters will appear for editing.


rs30-0802t1t1sdaehh [3] (with I/O data)
Slot 0 (with I/O data) *
Interface 1 (PN-IO) *
Port 1 (Twisted Pair T/TX RJ45 (10/100/1000 Mbit)) *
Port 2 (Twisted Pair T/TX RJ45 (10/100/1000 Mbit)) *
Port 3 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 4 (Twisted Pair TX RJ45 (10/100 Mbit)) *
Port 5 (Twisted Pair TX RJ45 (10/100 Mbit)) *



Sub-Module Parameters (GSDML Tab)

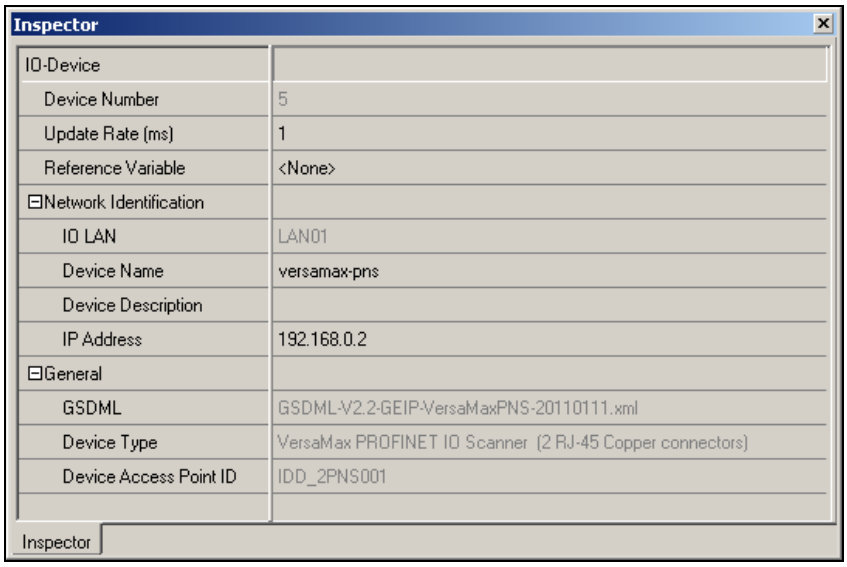
The GSDML tab displays the device's GSDML parameters, which cannot be edited.


Port parameter
GSDML Details
Port: Twisted Pair T/TX RJ45 (10/100/1000 Mbit)
Port: Twisted Pair T/TX RJ45 (10/100/1000 Mbit)
MAU Types: 10 11 15 16 29 30
Maximum Tx Delay: 1200
Maximum Rx Delay: 333
Deactivation Supported: True
Link State Diagnosis Capability: UpDown
Power Budget Control Supported: False
Is Default Ring Port: True

4.2.7 *Viewing / Editing IO-Device Properties*

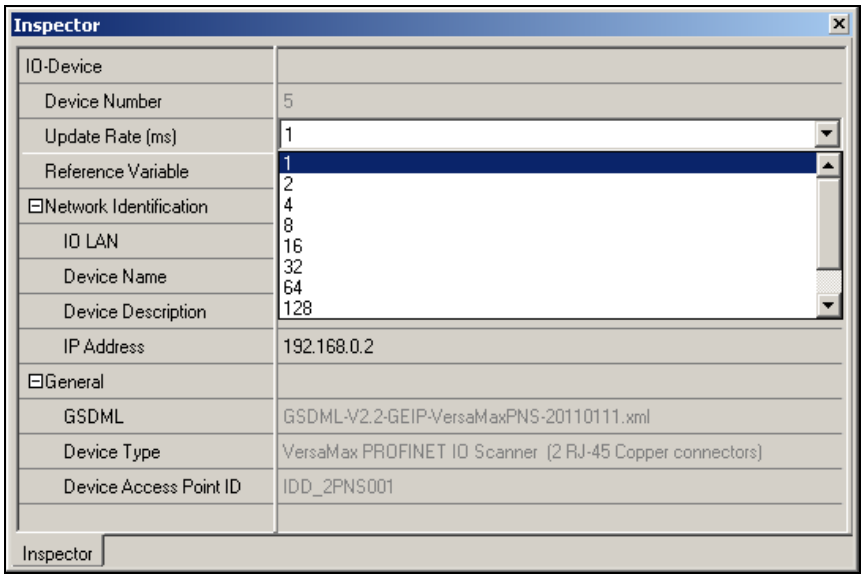
In addition to the parameter configuration described on the previous pages, all IO-Devices (PNS modules and third-party devices) have other configurable properties. These properties are displayed in the Proficy Machine Edition Inspector pane when the device is selected in the Navigator.

In the Inspector, properties that are not grayed-out can be used as is or edited as appropriate.



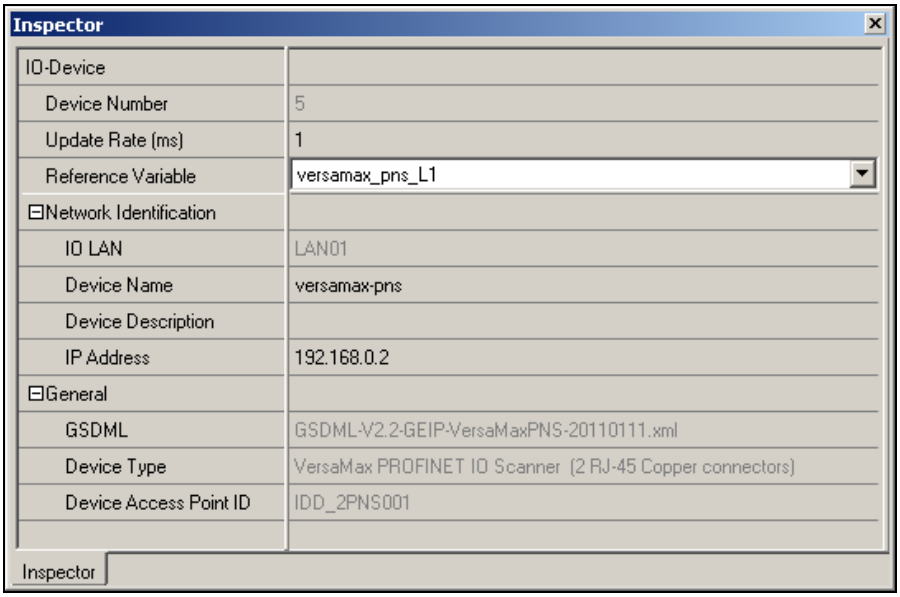
Device Number: A number automatically assigned to the device in the configuration.

Update Rate: The period between PROFINET cyclic data transfers between an IO-Controller and an IO-Device. It defaults to 32ms for VersaMax PNS modules, and 128ms for other devices. To change the update period of the PROFINET production cycle for the IO-Device, use the drop-down list to select 1, 2, 4, 8, 16, 32, 64, 128, 256, or 512ms:



Note: Because the update rate affects loading on the PNC, the maximum number of devices (up to 128) is limited to the equivalent of eight devices with 1ms update rates. For additional information, see “PNC Loading Limits” on page 48.

Reference Variable: To be used by the PNIO_DEV_COMM logic blocks. The choice defaults to none. To create a reference variable for the device, use the drop-down list to select Create. The variable name appears in the Inspector field:



IO LAN: (Read-only) Identifies the LAN of which the IO-Device is a part.

Device Name: This can be edited within the Inspector, or the default name can be used. Space characters are not permitted.

Device Description: An optional description can be entered for the IO-Device.

IP Address: IP address for the IO-Device. Default is assigned the lowest value that is currently available within the automatic IP address range defined for the LAN on which the device resides.

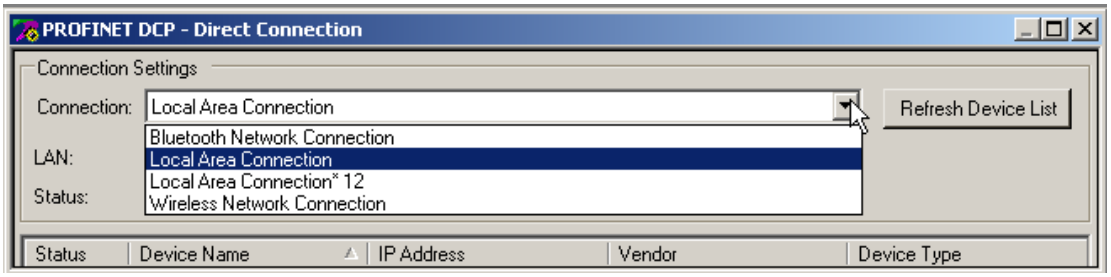
4.2.8 Assigning IO-Device Names

After the PNS and third-party IO-Devices on the LAN have been entered into the configuration, the Discovery and Configuration Protocol (DCP) tool in PME must be used to assign a name to each IO-Device. This step is required before downloading the configuration from the PNC, or the PNC will be unable to connect to the devices and deliver their configuration.

The programmer must be connected to the RXi Controller system and the LAN and its devices must be installed.

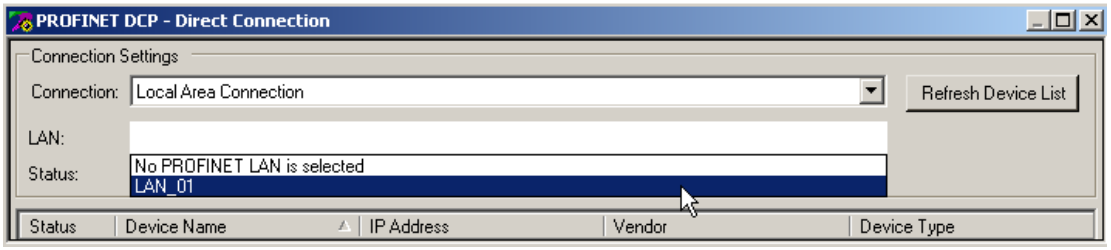
To open the DCP tool, right-click the hardware configuration containing the PROFINET Controller and choose Launch Discovery Tool from the menu.

Use the Connection dropdown list to select the computer port being used by the programmer to communicate with the RXi system:



The choices should match the windows network setting in the computer's network control panel.

In the DCP tool, use the LAN list to select the configured LAN to validate the results against.

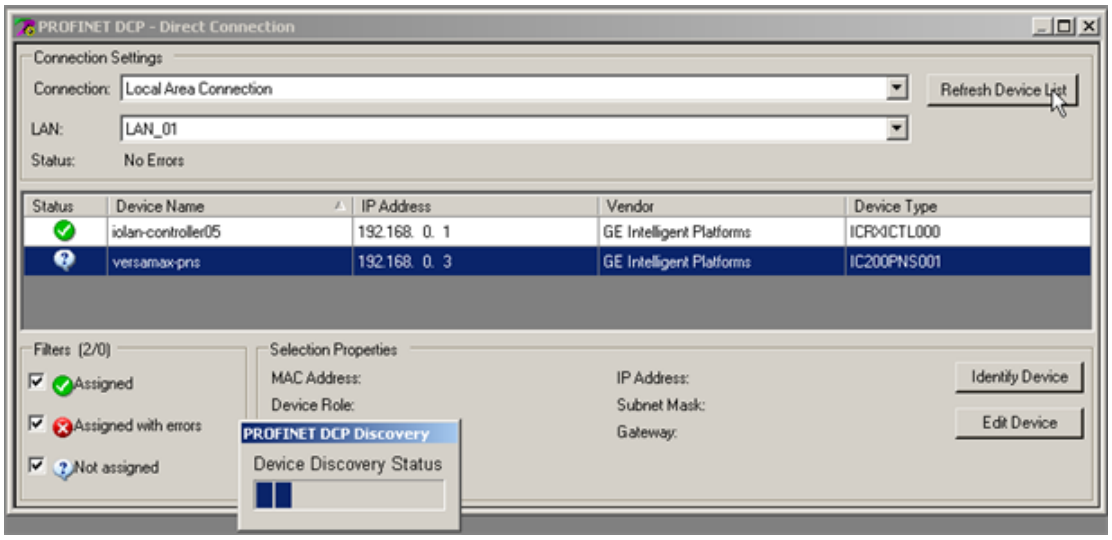


Once a LAN is selected, click Refresh Device List to display a list of actual devices on the LAN. Each device can be in one of three states as indicated by the symbol in the Status column:

indicates that the actual Device's Name, IP address, subnet, gateway matches the configured Device of the same Name residing on the selected LAN.

indicates that the actual Device's Name matches, but the IP address, subnet, or gateway doesn't match the configured Device of the same Name residing on the selected LAN.

indicates that the actual Device's Name does not match any configured Device residing on the selected LAN. Either there is no device name configured in the system, or that name is on a different LAN.



If the name of the device on the network is not correct, you must update the device name. To do this, select the device in the list of devices and click Edit Device. This will open a new dialog that can be used to set various parameters, including device name, directly on the device.

Note: Only the device name is required to match configuration in order for the PNC to successfully deliver configuration to the device.

4.2.9 After the Configuration is Stored to the RXi CPU

If the configuration is stored to non-volatile memory, the RXi Controller maintains configuration data over a power cycle.

The PNC transfers the configuration for remote IO-Devices over the PROFINET network. PROFINET delivers IO-Device configurations when the IO-Controller establishes an Application Relationship (AR) with the IO-Device. If all Application Relationships (AR)s are lost, the IO-Device does not change the configuration of any of its I/O sub-modules. Each sub-module retains the most recent configuration received since it was last powered up or restarted. If a sub-module has not been configured since powerup or restart, it remains in its hardware default condition. When the AR(s) are re-established and a configuration is sent to the IO-Device, and the configuration of a sub-module has changed, the IO-Device applies the new configuration.

If the PNC cannot connect to an IO-Device, the PNC logs a Loss of Device fault to the Controller Fault tables. The PNC periodically attempts to establish communications and configure the IO-Device. When one of these subsequent connect/configuration attempts is successful, the PNC logs an Addition of Device fault for that IO-Device in the Controller Fault tables.

Note: It may take up to 5–10 seconds for the PNC to establish a connection to an IO-Device, including one that previously existed, but was lost.

Whether or not an IO-Device sets its outputs to their defaults when it receives a configuration depends on the type of device. Refer to the device manufacturer's documentation to determine this behavior.

If a PNS is powered up after receiving a configuration and the connection is then lost, configuration defaults are applied when the connection is lost. On a subsequent connection, if the same configuration is sent, outputs remain at the configuration defaults until the first output data arrives. If a different configuration is sent, output defaults transition to the new defaults until the first output data arrives.

Clearing the RXi Controller Configuration

If the programmer or IDM clears an RXi Controller's configuration, only the non-volatile PNC parameters are retained. For a list of these parameters, see "4.2.3.5, Non-Volatile Configuration Parameters and Properties for PNC."

When the configuration is cleared, all open PROFINET IO-Device connections are closed. IO-Devices react to clearing the PNC as a loss of connectivity and take appropriate actions such as defaulting outputs.

4.3 Configuring the Embedded Ethernet Interface

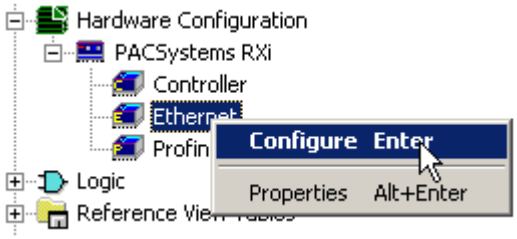
Before you can use the embedded Ethernet Interface, you must configure it using the programming software.

Ethernet interface configuration includes the following procedures:

- Using the IDM to assign a temporary IP address (page 33) for initial network operation, such as connecting the programmer to download the hardware configuration.
- Configuring the characteristics of the Ethernet interface.

To configure the embedded Ethernet interface:

1. Right click the Ethernet interface to display its parameters: IP Address, Subnet Mask and Gateway IP Address. Consult your network administrator for the proper values for these parameters.



2. Go online with the target and download the configuration through the embedded Ethernet port, using the factory-loaded default IP address which is 192.168.0.100. This address is intended only for initial connection in order to complete the configuration. (To set the IP address PME will use to connect to the RXi, edit the IP Address setting in the target properties.)

4.3.1 Ethernet Interface Configuration Parameters

Configuration Mode: This is fixed as TCP/IP.

IP Addresses: These values should be assigned by the person in charge of your network (the network administrator). TCP/IP network administrators are familiar with these parameters. It is important that these parameters are correct, otherwise the Ethernet Interface may be unable to communicate on the network and/or network operation may be corrupted. It is especially important that each node on the network is assigned a *unique* IP address.

If you have no network administrator and are using a simple *isolated network* with no gateways, you can use the following range of values for the assignment of local IP addresses:

10.0.0.1	First Ethernet interface
10.0.0.2	Second Ethernet interface
10.0.0.3	Third Ethernet interface
.	.
.	.
.	.
10.0.0.255	Programmer TCP or host

Also, in this case, set the subnet mask to 255.0.0.0 and the gateway IP address to 0.0.0.0.

Note: If the isolated network is connected to another network, the IP addresses 10.0.0.1 through 10.0.0.255 must not be used and the subnet mask, and gateway IP address must be assigned by the network administrator. The IP addresses must be assigned so that they are compatible with the connected network.

Status Address: The Status Address is the reference memory location for the Ethernet Interface status data. The Ethernet Interface automatically maintains 16 LAN Interface Status (LIS) bits in this location. The Status address can be assigned to valid %I, %Q, %R, %AI, %AQ or %W memory. The default value is the next available %I address.

Note: Do not use the 80 bits configured as Ethernet Status data for other purposes or data will be overwritten.

Note: If the Ethernet interface's Variable Mode property is set to true, the Status Address parameter is removed from the Settings tab. Instead, Ethernet Status references must be defined as I/O variables on the Terminals tab.

Length: This is the total length of the Ethernet Interface status data. This is automatically set to either 80 bits (for %I and %Q Status address locations) or 5 words (for %R, %AI, %AQ and %W Status address locations).

For definitions of the Ethernet interface Status bits, refer to 9.8.1.3, Ethernet Interface Status Bits."

4.3.2 Pinging TCP/IP Ethernet Interfaces on the Network

PING (Packet InterNet Grouper) is the name of a program used on TCP/IP networks to test reachability of destinations by sending them an ICMP echo request message and waiting for a reply. Most nodes on TCP/IP networks, including the PACSystems Ethernet Interface, implement a PING command.

You should ping each installed Ethernet Interface. When the Ethernet Interface responds to the ping, it verifies that the interface is operational and configured properly. Specifically it verifies that acceptable TCP/IP configuration information has been downloaded to the Interface.

4.3.2.1 Determining if an IP Address is Already Being Used

Note: This method does not guarantee that an IP address is not duplicated. It will not detect a device that is configured with the same IP address if it is temporarily off the network.

It is very important not to duplicate IP addresses. To determine if another node on the network is using the same IP address:

1. Disconnect your Ethernet Interface from the LAN.
2. Ping the disconnected Interface's IP address. If you get an answer to the ping, the chosen IP address is already in use by another node. You *must* correct this situation by assigning a unique IP address.

4.3.3 Terminals Tab

This configuration tab is displayed only when the Ethernet interface's Variable Mode property is set to True. When Variable Mode is selected, the Ethernet Status bits are referenced as I/O variables that are mapped to the Ethernet status bits on this configuration tab.

The use of I/O variables allows you to configure the Ethernet interface without having to specify the reference addresses to use for the status information. Instead, you can directly associate variable names with the status bits. For more information, refer to "7.1.3, I/O Variables."

4.4 Storing (Downloading) Hardware Configuration

A PACSystems control system is configured by creating a configuration file in the programming software, then transferring (downloading) the file from the programmer to the controller.

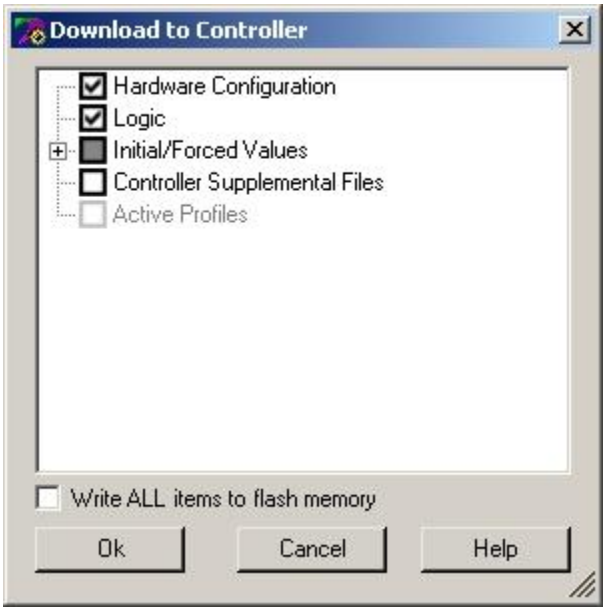
The CPU stores the configuration file in its non-volatile memory. After the configuration is stored, I/O scanning is enabled or disabled according to the newly stored configuration parameter, Stop-Mode I/O.

Before you can store the hardware configuration to the RXi Controller, you must first set the IP address in the Ethernet Interface using the IDM (page 33.)

1. In the programmer software, go to the Project tab of the Navigator, right click the Target, choose Set as Active Target and then Go Online.
2. Right click the Target and choose Online Commands, Set Programmer Mode. Make sure the CPU is in Stop mode.
3. Right click the Target node, and choose Download to Controller.
4. In the Download to Controller dialog box, select the items to download and click OK.

Note: If you download to a PACSystems target that already has a project on it, the existing project is overwritten.

Note: If hardware configuration and logic are coupled they cannot be stored independently. They must be stored at the same time.



Chapter 5. CPU Operation

This chapter describes the operating modes of a PACSystems RXi CPU and describes the tasks the CPU carries out during these modes. The following topics are discussed:

- CPU Sweep
- Program Scheduling Modes
- Window Modes
- Run/Stop Operations
- Flash Memory Operation
- Clocks and Timers
- System Security
- I/O System
- Power-Up and Power-Down Sequences

5.1 CPU Sweep

The application program in the CPU executes repeatedly until stopped by a command from the programmer, from another device, from the Intelligent Display Module, or a fatal fault occurs. In addition to executing the application program, the CPU obtains data from input devices, sends data to output devices, performs internal housekeeping, performs communications tasks, and performs self-tests. This sequence of operations is called the ***sweep***.

The CPU sweep runs in one of three sweep modes:

Normal Sweep	In this mode, each sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The Communications and Background Windows can be set to execute in Limited or Run-to-Completion mode.
Constant Sweep	In this mode, each sweep begins at a user-specified Constant Sweep time after the previous sweep began. The Logic Window is executed in its entirety each sweep. If there is sufficient time at the end of the sweep, the CPU alternates among the Communications and Background Windows, allowing them to execute until it is time for the next sweep to begin.
Constant Window	In this mode, each sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The CPU alternates among the Communications and Background Windows, allowing them to execute for a time equal to the user-specified Constant Window timer.

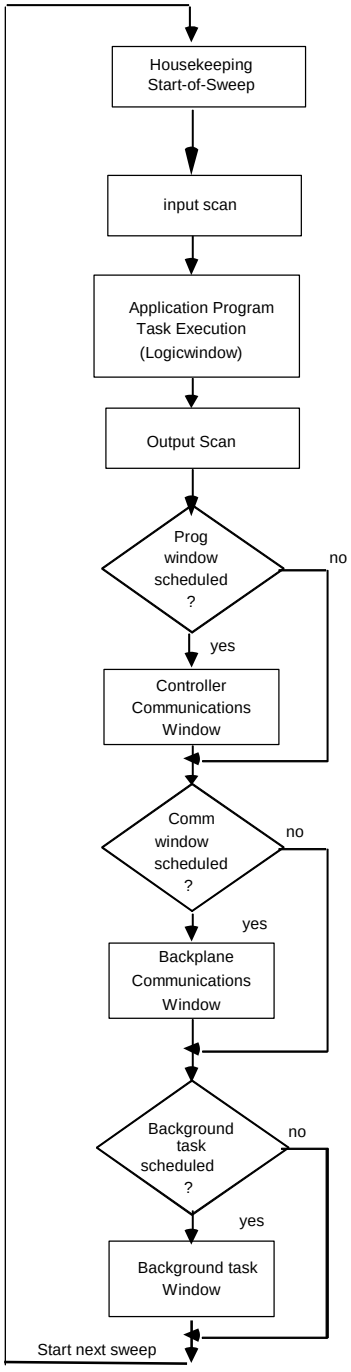
Note: The information presented above summarizes the different sweep modes. For additional information on each mode, refer to “5.1.2, CPU Sweep Modes.”

The CPU also operates in one of four Run/Stop Modes (for details, see “5.5, Run/Stop Operations”):

- Run/Outputs Enabled
- Run/Outputs Disabled
- Stop/IO Scan
- Stop/No IO

5.1.1 Parts of the CPU Sweep

The major phases in a typical CPU sweep are shown in the following figure.



Parts of a Typical CPU Sweep

Major Phases in a Typical CPU Sweep

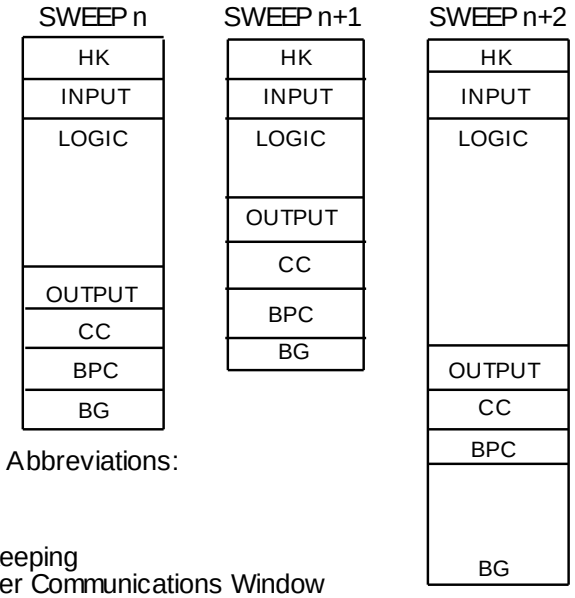
Phase	*Activity*
Housekeeping	The housekeeping portion of the sweep performs the tasks necessary to prepare for the start of the sweep. This includes updating %S bits, determining timer update values and determining the mode of the sweep (Stop or Run.
Input Scan	During the input scan, the CPU reads input data from the embedded PNC. **Note:** The input scan is not performed if a program has an active Suspend I/O function on the previous sweep.
Application Program Task Execution (Logic Window)	The CPU solves the application program logic. It always starts with the first instruction in the program. It ends when the last instruction is executed. Solving the logic creates a new set of output data. Interrupt driven logic can execute during any phase of the sweep. For details on controlling the execution of programs, refer to “Chapter 6, Program Organization.” A list of execution times for instructions can be found in “Appendix A, CPU Performance Data.”
Output Scan	The CPU writes output data to the embedded PNC. If the CPU is in Run mode and it is configured to perform a background checksum calculation, the background checksum is performed at the end of the output scan. The default setting for number of words to checksum each sweep is 16. If the words to checksum each sweep is set to zero, this processing is skipped. The background checksum helps ensure the integrity of the user logic while the CPU is in Run mode. The output scan is not performed if a program has an active Suspend I/O function on the current sweep.
Controller Communications Window	Communications on the Ethernet ports are serviced in this window. The CPU always executes this window. Time and execution of the Controller Communications Window can be configured using the programming software. It can also be dynamically controlled from the user program using Service Request function #3. The window time can be set to a value from 0 to 255 milliseconds (default is 10 milliseconds). If the Controller Communications Window is set to 0, there are two ways to open the window: perform a power-cycle or go to Stop mode.
Backplane Communications Window	Even though the RXi Controller does not operate on a backplane, the Backplane Communications window still runs and parameters that affect it are implemented. The Backplane Communications Window defaults to Complete (Run to Completion) mode. This window can also run in Limited mode, in which the maximum time allocated for the window per scan is specified. The mode and time limit can be configured and stored to the CPU, or it can be dynamically controlled from the user program using Service Request function #4. The Backplane Communications Window time can be set to a value from 0 to 255ms (default is 255ms). This allows communications functions to be skipped during certain time-critical sweeps.
Background Window	CPU self-tests occur in this window. A CPU self-test is performed in this window. Included in this self-test is a verification of the checksum for the CPU operating system software. The Background Window time defaults to 0 milliseconds. A different value can be configured and stored to the CPU, or it can be changed online using the programming software. Time and execution of the Background Window can also be dynamically controlled from the user program using Service Request function #5. This allows background functions to be skipped during certain time-critical sweeps.

5.1.2 CPU Sweep Modes

5.1.2.1 Normal Sweep Mode

In Normal Sweep mode, each sweep can consume a variable amount of time. The Logic window is executed in its entirety each sweep. The Communications window can be set to execute in a Limited or Run-to-Completion mode. Normal Sweep is the most common sweep mode used for control system applications.

The following figure illustrates three successive CPU sweeps in Normal Sweep mode. Note that the total sweep times may vary due to sweep-to-sweep variations in the Logic, Communications and Background windows.



Abbreviations:

HK = Housekeeping
CC = Controller Communications Window
BPC = Backplane Communications Window
BG = Background Window

Typical Sweeps in Normal Sweep Mode

5.1.2.2 Constant Sweep Mode

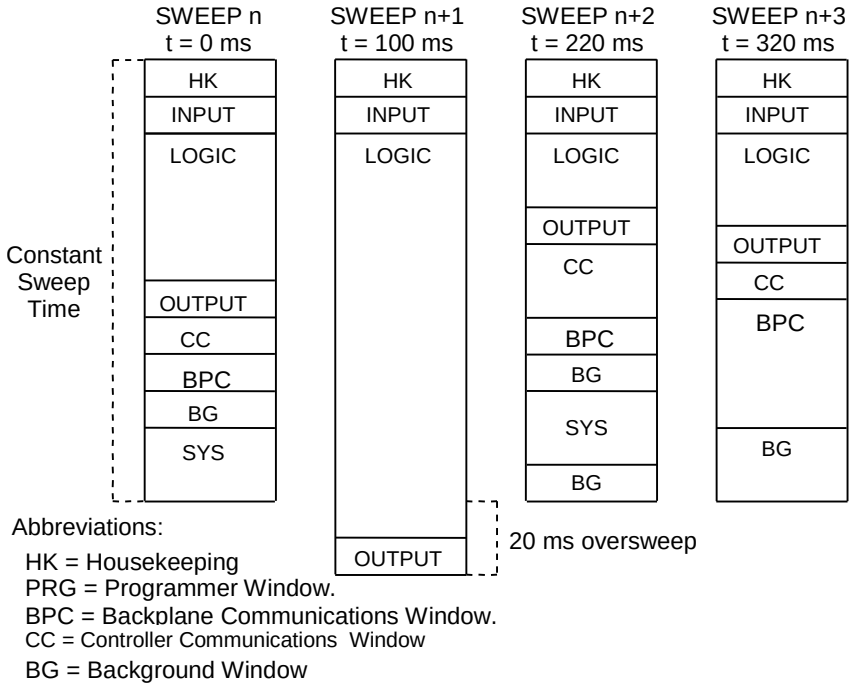
In Constant Sweep mode, each sweep begins at a specified Constant Sweep time after the previous sweep began. The Logic Window is executed in its entirety each sweep. If there is sufficient time at the end of the sweep, the CPU alternates among the Controller Communications, Backplane Communications, and Background windows, allowing them to execute until it is time for the next sweep to begin. Some or all of the Communications and Background Windows may not be executed. The Communications and Background Windows terminate when the overall CPU sweep time has reached the value specified as the Constant Sweep time.

One reason for using Constant Sweep mode is to ensure that I/O data are updated at constant intervals.

The value of the Constant Sweep timer can be configured to be any value from 5 to 2550 milliseconds. The Constant Sweep timer value may also be set and Constant Sweep mode may be enabled or disabled by the programming software or by the user program using Service Request function #1. The Constant Sweep timer has no default value; a timer value must be set prior to or at the same time Constant Sweep mode is enabled.

If the sweep exceeds the Constant Sweep time in a given sweep, the CPU places an oversweep alarm in the Controller Fault table and sets the OV_SWP (%SA0002) status reference at the beginning of the next sweep. Additional sweep time due to an oversweep condition in a given sweep does not affect the time given to the next sweep.

The following figure illustrates four successive sweeps in Constant Sweep mode with a Constant Sweep time of 100 milliseconds. Note that the total sweep time is constant, but an oversweep may occur due to the Logic Window taking longer than normal.



Typical Sweeps in Constant Sweep Mode

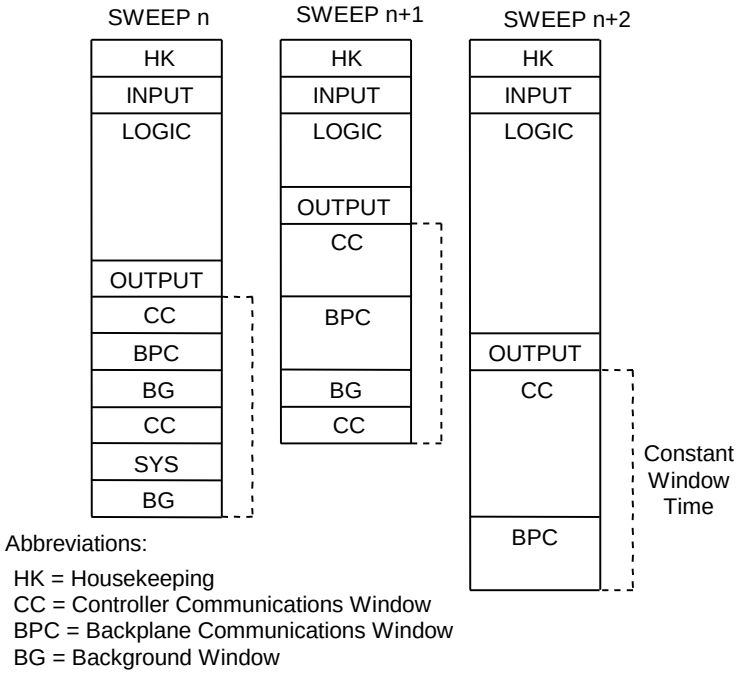
5.1.2.3 Constant Window Mode

In Constant Window mode, each sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The CPU alternates among the three windows, allowing them execute for a time equal to the value set for the Constant Window timer. The overall CPU sweep time is equal to the time required to execute the Housekeeping, Input Scan, Logic Window, and Output Scan phases of the sweep plus the value of the Constant Window timer. This time may vary due to sweep-to-sweep variances in the execution time of the Logic Window.

Constant Sweep mode could be used for an application that requires a certain amount of time between the Output Scan and the Input Scan, permitting inputs to settle after receiving output data from the program.

The value of the Constant Window timer can be any value from 3 to 255 milliseconds, and can be set by the programming software or by the user program using Service Request functions #3, #4, and #5.

The following figure illustrates three successive sweeps in Constant Window mode. Note that the total sweep times may vary due to sweep-to-sweep variations in the Logic Window, but the time given to the Communications and Background windows is constant. Some Communications or Background Windows may be skipped, suspended, or run multiple times based on the Constant Window time.



Typical Sweeps in Constant Window Mode

5.2 Program Scheduling Modes

The CPU supports one program scheduling mode, the Ordered mode. An ordered program is executed in its entirety once per sweep in the Logic Window.

5.3 Window Modes

The previous section describes the phases of a typical CPU sweep. The Controller Communications, Backplane Communications, and Background windows can be run in various modes, based on the CPU sweep mode. (CPU sweep modes are described in detail on page 74.) The following three window modes are available:

Run-to-Completion	In Run-to-Completion mode, all requests made when the window has started are serviced. When all pending requests in the given window have completed, the CPU transitions to the next phase of the sweep. (This does not apply to the Background window because it does not process requests.)
Constant	In Constant Window mode, the total amount of time that the Controller Communications window, Backplane Communications window, and Background window run is fixed. If the time expires while in the middle of servicing a request, these windows are closed, and communications will be resumed the next sweep. If no requests are pending in this window, the CPU cycles through these windows the specified amount of time polling for further requests. If any window is put in constant window mode, all are in constant window mode.
Limited	In Limited mode, the maximum time that the window runs is fixed. If time expires while in the middle of servicing a request, the window is closed, and communications will be resumed the next time that the given window is run. If no requests are pending in this window, the CPU proceeds to the next phase of the sweep.

5.4 Data Coherency in Communications Windows

When running in Constant or Limited Window mode, the Controller and Backplane Communications Windows may be terminated early in all CPU sweep modes. If an external device, such as CIMPLICITY HMI, is transferring a block of data, the coherency of the data block may be disrupted if the communications window is terminated prior to completing the request. The request will complete during the next sweep; however, part of the data will have resulted from one sweep and the remainder will be from the following sweep. When the CPU is in Normal Sweep mode and the Communications Window is in Run-to-Completion mode, the data coherency problem described above does not exist.

Note: External devices that communicate to the CPU while it is stopped will read information as it was left in its last state. This may be misleading to operators viewing an HMI system that does not indicate CPU Run/Stop state. Process graphics will often indicate everything is still operating normally.

Also, note that non-retentive outputs do not clear until the CPU is changed from Stop to Run.

5.5 Run/Stop Operations

The PACSystems CPUs support four run/stop modes of operation. You can change these modes in the following ways: configuration from the programming software, LD function blocks, and system calls from C applications. Switching to and from various modes can be restricted based on privilege levels, passwords, etc.

Mode	Operation
Run/Outputs Enabled	The CPU runs user programs and continually scans inputs and outputs. The Controller and Backplane Communications Windows are run in Limited, Run-to-Completion, or Constant mode.
Run/Outputs Disabled	The CPU runs user programs and continually scans inputs, but updates to outputs are not performed. Outputs are held in their configured default state in this mode. The Controller and Backplane Communications Window are run in Limited, Run-to-Completion, or Constant mode.
Stop/IO Scan Enabled	The CPU does not run user programs, but the inputs and outputs are scanned. The Controller and Backplane Communications Windows are run in Run-to-Completion mode. The Background Window is limited to 10 ms.
Stop/IO Scan Disabled	The CPU does not run user programs, and the inputs and outputs are not scanned. The Controller and Backplane Communications Windows are run in a Run-to-Completion mode. The Background Window is limited to 10 ms.

Note: You cannot add to the size of %P and %L reference tables in Run Mode unless the %P and %L references are the first of their type in the block being stored or the block being stored is a totally new block.

5.5.1 CPU Stop Modes

The CPU has two modes of operation while it is in Stop mode:

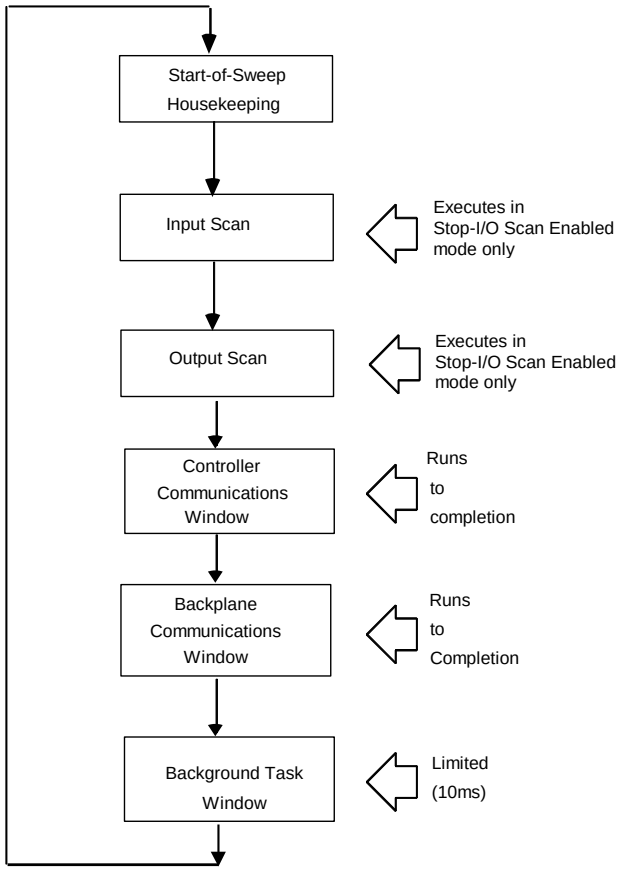
- I/O Scan Enabled - the Input and Output scans are performed each sweep
- I/O Scan Disabled - the Input and Output scans are skipped

When the CPU is in Stop mode, it does not execute the application program. You can configure whether the I/O is scanned during Stop mode. Communications with the programmer and intelligent option modules continue in Stop mode.

In both Stop modes, the Controller Communications and Backplane Communications windows run in Run-to-Completion mode and the Background window runs in Limited mode with a 10 millisecond limit.

The number of last scans can be configured in the hardware configuration. Last scans are completed after the CPU has received an indication that a transition from Run to Stop or Stop Faulted mode should occur. The default is 0.

The application program can use SVC_REQ13 to stop the CPU after a specified number of scans. All I/O will go to their configured default states, and a diagnostic message will be placed in the Controller Fault Table.



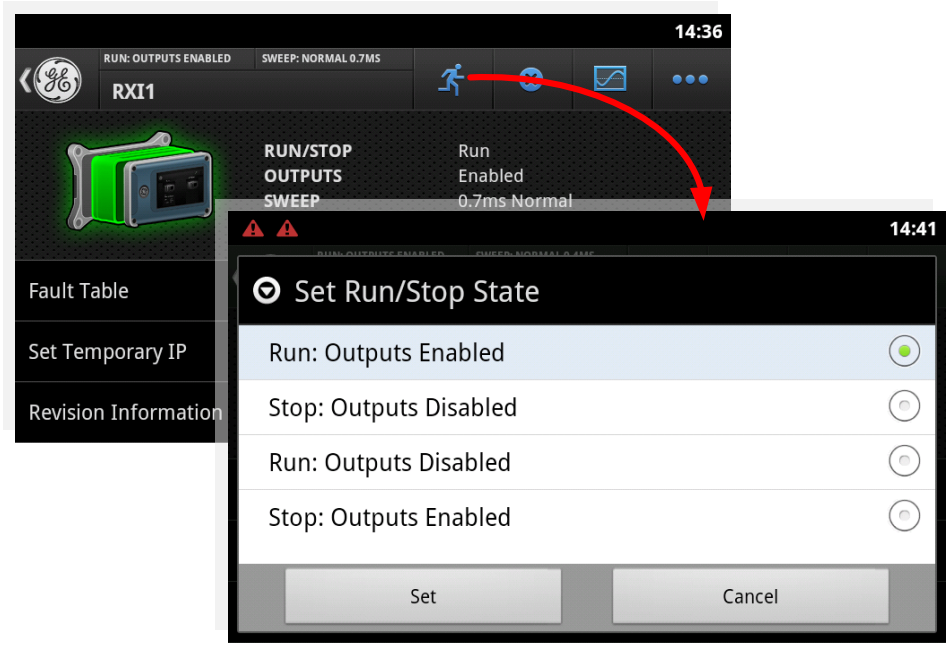
CPU Sweep in Stop- I/O Disabled and Stop- I/O Enabled Modes

5.5.2 Stop-to-Run Mode Transition

The CPU performs the following operations on Stop-to-Run transition:

- Validation of sweep mode and program scheduling mode selections
- Validation of references used by programs with the actual configured sizes
- Re-initialization of data areas for external blocks and standalone C programs
- Clearing of non-retentive memory

Note: You can use the IDM to change the RXi Controller's run/stop state as follows.



5.6 Flash Memory Operation

The CPU stores the current configuration and application in user memory. You can also store this data into non-volatile flash memory. The PACSystems CPU provides enough flash memory to hold all of user space, all reference tables that aren't counted against user space, and any overhead required. For details on which items count against user memory space, refer to Appendix C.

By default, the CPU reads program logic and configuration, and reference table data from user memory at powerup. However, logic/configuration and reference tables can each be configured to always read from flash or conditionally read from flash. To configure these parameters in the programming software, select the CPU's Settings tab in Hardware Configuration.

If logic/configuration and/or reference tables are configured for conditional powerup from flash, these items are restored from flash to user memory when user memory is corrupted or not preserved. When user memory is preserved, no flash operation occurs."

If logic/configuration and/or reference tables are configured to always power up from flash, these items are restored from flash to user memory regardless of the state of the user memory.

Note: If ***any*** component (logic/configuration or reference tables) is read from flash, OEM-mode and passwords are also read from flash.

In addition to configuring where the CPU obtains logic, configuration, and data during powerup, the programming software provides the following flash operations:

- Write a copy of the current configuration, application program, and reference tables (excluding overrides) to flash memory. Note that a write-to-flash operation causes all components to be stored to flash.
- Read a previously stored configuration and application program, and/or reference table values from flash into user memory.
- Verify that flash and user memory contain identical data.
- Clear flash contents.

Flash read and write operations copy the contents of flash memory or user memory as individual files. The programming software displays the progress of the copy operation and allows you to cancel a flash read or write operation during the copy process instead of waiting for the entire transfer process to complete. The entire user memory image must be successfully transferred for the flash copy to be considered successful. If an entire write-to-flash transfer is not completed due to canceling, power cycle, or some other intervention, the CPU will clear flash memory. Similarly, if a read-from-flash transfer is interrupted, user memory will be cleared.

5.7 Logic/Configuration Source and CPU Operating Mode at Power-up

Flash and user memory can contain different values for the Logic/Configuration Power-up Source parameter. The following tables summarize how these settings determine the logic/configuration source after a power cycle. CPU mode is affected by the Power-up Mode and Stop-Mode I/O Scanning parameters, and the power down mode as shown in the tables on page 83.

Before Power Cycle		After Power Cycle	
Logic/Configuration Power-up Source in Flash	**Logic/Configuration Power-up Source in RAM**	**Origin of Logic/Configuration**	**CPU Mode**
Always Flash	Memory not preserved (i.e. memory corrupted)	Flash	See “CPU Mode when Memory is not Preserved and Power-up Source is Flash” on page 83.
Always Flash	No configuration in RAM, memory preserved	Flash	See” Memory Preserved” on page 83.
Always Flash	Always Flash	Flash	
Always Flash	Conditional Flash	Flash	
Always Flash	Always RAM	Flash	
Conditional Flash	Memory not preserved (i.e. memory corrupted)	Flash	See “CPU Mode when Memory is not Preserved and Power-up Source is Flash” on page 83.
Conditional Flash	No configuration in RAM, memory preserved	Uses default logic/configuration	Stop Disabled
Conditional Flash	Always Flash	RAM	See ”CPU Mode when Memory is Preserved” on page 83.
Conditional Flash	Conditional Flash	RAM	
Conditional Flash	Always RAM	RAM	
Always RAM	Memory not preserved (i.e. memory corrupted)	Uses default logic/configuration	Stop Disabled
Always RAM	No configuration in RAM, memory preserved	Uses default logic/configuration	Stop Disabled
Always RAM	Always Flash	Flash	See ”CPU Mode when Memory is Preserved” on page 83.
Always RAM	Conditional Flash	RAM	
Always RAM	Always RAM	RAM	
No Configuration in Flash	Memory not preserved (i.e. memory corrupted)	Uses default logic/configuration	Stop Disabled
No Configuration in Flash	No configuration in RAM, memory preserved	Uses default logic/configuration	Stop Disabled
No Configuration in Flash	Always Flash	RAM	See ”CPU Mode when Memory is Preserved” on page 83.
No Configuration in Flash	Conditional Flash	RAM	
No Configuration in Flash	Always RAM	RAM	

CPU Mode when Memory is not Preserved and Power-up Source is Flash

Configured Power-up Mode	CPU Mode
Run	Run Enabled
Stop	Stop Disabled
Last	Run Disabled

CPU Mode when Memory is Preserved

Configuration Parameters		Power Down Mode	CPU Mode
Power-up Mode	Stop-Mode I/O Scanning		
Run	N/A	N/A	Run Enabled
Stop	Enabled	N/A	Stop Enabled
Stop	Disabled	N/A	Stop Disabled
Last	N/A	Stop Disabled	Stop Disabled
Last	Enabled	Stop Enabled	Stop Enabled
Last	Disabled	Stop Enabled	Stop Disabled
Last	N/A	Run Disabled	Run Disabled
Last	N/A	Run Enabled	Run Enabled

5.8 Clocks and Timers

Clocks and timers provided by the CPU include an elapsed time clock, a time-of-day clock, and software and hardware watchdog timers.

For information on timer instructions provided by the CPU instruction set, refer to the Proficy Machine Edition online help.

5.8.1 Elapsed Time Clock

The elapsed time clock tracks the time elapsed since the CPU powered on. The clock is not retentive across a power failure; it restarts on each power-up. This seconds count rolls over (seconds count returns to zero) approximately 100 years after the clock begins timing.

Because the elapsed time clock provides the base for system software operations and timer function blocks, it may not be reset from the user program or the programmer. However, the application program can read the current value of the elapsed time clock by using Service Request #16 or Service Request #50, which provides higher resolution.

5.8.2 Time of Day Clock

A hardware real time clock (RTC) maintains the time of day (TOD) in the CPU. The time-of-day clock maintains the following time functions:

- Year (two digits)
- Month
- Day of month
- Hour
- Minute
- Second
- Day of week

The TOD clock is battery-backed and maintains its present state across a power failure. The time-of-day clock handles month-to-month and year-to-year transitions and automatically compensates for leap years through year 2036.

You can read and set the hardware TOD time and date through the application program using Service Request function #7.

5.8.2.1 High-Resolution Time of Day Software Clock

A high-resolution software TOD clock is implemented in firmware to provide nanoseconds resolution. When the high-resolution software TOD clock is set, the hardware TOD clock is set with the YYYY: Mon: Day: Hr: Min: Sec fields in the POSIX time, the RTC is read, and the delta between the POSIX time and the value read from the RTC is computed and saved. Thus, if 1-second resolution is desired the hardware TOD clock is read. Otherwise, the high-resolution software TOD clock is read to provide greater resolution. When the latter occurs, the hardware RTC is read and the saved delta added to the value read.

When the SNTP Time Transfer feature is implemented, all SNTP time updates received at the CPU shall update the high-resolution software TOD clock.

5.8.3 Watchdog Timer

5.8.3.1 Software Watchdog Timer

A software watchdog timer in the CPU is designed to detect “failure to complete sweep” conditions. The timer value for the software watchdog timer is set using the programming software. The allowable range for this timer is 10 to 2550 milliseconds; the default value is 200 milliseconds. The software watchdog timer always starts from zero at the beginning of each sweep.

The software watchdog timer is useful in detecting abnormal operation of the application program that prevents the CPU sweep from completing within the user-specified time. Examples of such abnormal application program conditions are as follows:

- Excessive recursive calling of a block
- Excessive looping (large loop count or large amounts of execution time for each iteration)
- Infinite execution loop

When selecting a software watchdog value, always set the value higher than the longest expected sweep time to prevent accidental expiration. For Constant Sweep mode, allowance for oversweep conditions should be considered when selecting the software watchdog timer value.

The watchdog timer continues during interrupt execution. Queuing of interrupts within a single sweep may cause watchdog timer expiration.

If the software watchdog timeout value is exceeded, the CPU goes to Stop-Halt mode. A fault is placed in the Controller Fault table and outputs go to their default state. The CPU will communicate with the programmer and the IDM; no other communications or operations are possible. To recover, cycle power.

To extend the current sweep beyond the software watchdog timer value, the application program can restart the software watchdog timer using Service Request function #8. However, the software watchdog timer value may only be changed from the configuration software.

The programmer can connect to the CPU in Stop-Halt mode through the embedded Ethernet port without a reset or power cycle.

5.8.3.2 Hardware Watchdog Timer

A backup circuit provides additional protection for the CPU. If this backup circuit activates, the CPU is immediately placed in Reset mode. Outputs go to their default state and no communications of any form are possible, and the CPU will halt.

CPU Response to a Hardware Watchdog Timeout:

- The CPU automatically restarts and goes into Stop-Halt mode.
- The CPU retains fault tables after a hardware watchdog timeout.
- While the CPU is in Stop-Halt mode, you can connect the programmer software or PACs Analyzer to view the fault tables, including all faults logged before the timeout. (The PACS Analyzer software can be downloaded from the Support website.)
- The CPU does not retain Controller and I/O Fault tables following recovery from the Stop-Halt state. To recover from Stop-Halt mode and return to normal operation, all non-volatile memory must be cleared. This can be done by power cycling the unit.
- During startup following hardware watchdog reset, the CPU logs an *informational fault* with Error Code 446, which indicates a watchdog auto-reset occurred.

Note: PACSystems does not support Fatal Fault Retries.

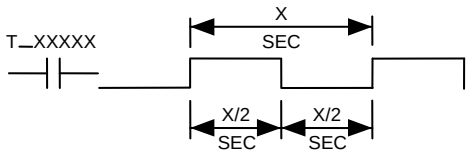
5.8.4 Timed Contacts

The PACSystems has four timed contacts that can be used to provide regular pulses of power flow to other program functions. Timed contacts cycle on and off, in square-wave form, every 0.01 second, 0.1 second, 1.0 second, and 1 minute. Timed contacts can be read by an external communications device to monitor the state of the CPU and the communications link. Timed contacts are also often used to blink pilot lights and LEDs.

The timed contacts are referenced as T_10MS (0.01 second), T_100MS (0.1 second), T_SEC (1.0 second), and T_MIN (1 minute). These contacts represent specific locations in %S memory:

#T_10MS	0.01 second timed contact	%S0003
#T_100MS	0.1 second timed contact	%S0004
#T_SEC	1.0 second timed contact	%S0005
#T_MIN	1.0 minute timed contact	%S0006

These contacts provide a pulse having an equal on and off time duration. The following timing diagram illustrates the on/off time duration of these contacts.



Caution

Do not use timed contacts for applications requiring accurate measurement of elapsed time. Timers, time-based subroutines, and PID blocks are preferred for these types of applications.

The CPU updates the timed contact references based on a free-running timer that has no relationship to the start of the CPU sweep. If the sweep time remains in phase with the timed contact clock, the contact will always appear to be in the same state. For example, if the CPU is in constant sweep mode with a sweep time setting of 100ms, the T_10MS and T_100MS bits will never toggle.

For additional information using the timed contacts instructions, refer to the Proficy Machine Edition online help.

5.9 System Security

The PACSystems CPU supports the following two types of system security:

- Passwords/privilege levels
- OEM protection

5.9.1 Passwords and Privilege Levels

Passwords are a configurable feature of the PACSystems CPU. Their use is optional and can be set up using the programming software. Passwords provide different levels of access privilege for the CPU.

The default state is no password protection. Each privilege level in the CPU may have a unique password; however, the same password can be used for more than one level. Passwords are one to seven ASCII characters in length. Passwords can be changed only through the programming software.

After passwords have been set up, access to the CPU via any communications path is restricted from the levels at which the passwords are set, unless the proper password has been entered. Once a password has successfully been accepted, access to the privilege level requested and below is granted (for example, providing the password for level 3 allows access to functions at levels 1, 2, and 3).

Note: The Run Mode selection on the Intelligent Display Module overrides password protection. Even though the programmer may not be able to switch between Run and Stop mode, the IDM can do so.

Privilege Levels

Level	*Password*	*Access Description*
4	Yes	Write to configuration or logic. Configuration may only be written in Stop mode; logic may be written in Stop or Run mode. Set or delete passwords for any level. ***Note:*** This is the default privilege for a connection to the CPU if no passwords are defined.
3	Yes	Write to configuration or logic when the CPU is in Stop mode, including word-for-word changes, addition/deletion of program logic, and the overriding of discrete I/O.
2	Yes	Write to any data memory. This does not include overriding discrete I/O. The CPU can be started or stopped. CPU and I/O fault tables can be cleared.
1	Yes	Read any CPU data, except for passwords. This includes reading fault tables, performing datagrams, verifying logic/configuration, loading program and configuration, etc. from the CPU. None of this data may be changed. At this level, transition to Run mode from the programmer is not allowed.

5.9.1.1 Protection Level Request from Programmer

Upon connection to the CPU, the programmer requests the CPU to move to the highest non-protected level.

The programmer requests a privilege level change by supplying the new privilege level and the password for that level. If the password sent by the programmer does not agree with the password stored in the CPU's password access table for the requested level, the privilege level change is denied and a fault is logged in the Controller Fault table. The current privilege level is maintained, and no change occurs. A request to change to a privilege level that is not password protected is made by supplying the new level and a null password. A privilege change may be to a lower level as well as to a higher level.

5.9.1.2 Disabling Passwords

The use of password protection is optional, unless Enhanced Security is enabled. To prevent the use of password protection, disable the feature in the programming software's hardware configuration.

Note: To enable passwords after they have been disabled, the CPU must be power-cycled.

Password protection prevents firmware upgrades. Before attempting a firmware upgrade, disable password protection, then enable it after the upgrade.

5.9.2 OEM Protection

OEM protection is similar to the passwords and privilege levels. However, OEM protection provides a higher level of security. The OEM protection feature is enabled/disabled using a 1 to 7 character password. When OEM protection is enabled, all read and write access to the CPU program and configuration is prohibited.

Protection for an OEMs' investment in software is provided in the form of a special password known as the *OEM key*. When the OEM key has been given a non-blank value, the CPU may be placed in a mode in which reads, writes, and verification of the logic and/or configuration are prohibited. This allows a third-party OEM to create Control Programs for the CPU and then set the OEM-locked mode, which prevents the end user from reading or modifying the program.

5.9.2.1 OEM Protection in Systems that Load from Flash Memory

For users that want the CPU to load from flash upon powerup, a special provision is made to activate OEM protection based on the OEM key stored in flash memory. If the OEM key that was stored to flash is non-blank, and the CPU is configured to load logic/configuration from flash, then upon powerup OEM protection is activated automatically. This is true even when OEM protection is not re-activated after the download

Users should be careful to record the OEM key for future reference if they are storing a non-blank OEM key to flash memory. If disabling OEM protection, be sure to clear the OEM key that is stored in flash memory.

5.9.2.2 OEM Protection and Firmware Upgrades

A firmware upgrade may be performed while a CPU is OEM protected. However, if a non-blank OEM key is stored to flash memory, and the CPU is configured to load logic/configuration from flash, then OEM protection remains active after the completion of a firmware upgrade.

5.10 PACSystems I/O System

The embedded PNC provides the interface between the CPU and other devices. For details, see “11.7, RXi CPU Operations for PROFINET.”

5.10.1 I/O System Diagnostic Data Collection

Diagnostic data in a PACSystems I/O system is obtained via diagnostic bits sent from the I/O modules to their PROFINET scanners. Diagnostic bits indicate the current fault status of the associated module. Bits are set when faults occur and are cleared when faults are cleared.

Diagnostic data is not maintained for modules from other manufacturers.

Note: At least two sweeps must occur to clear the diagnostic bits: one scan to send the %Q data to the module and one scan to return the %I data to the CPU. Because module processing is asynchronous to the controller sweep, more than two sweeps may be needed to clear the bits, depending on the sweep rate and the point at which the data is made available to the module.

5.11 Power-Up and Power-Down Sequences

5.11.1 Power-Up Sequence

System power-up consists of the following parts:

- Power-up self-test
- CPU memory validation
- System configuration
- I/O system initialization

5.11.1.1 Power-Up Self-Test

On system power-up, the CPU module executes hardware checks and software validity checks.

In the initial release, a failed battery fault is logged in the Controller fault table.

5.11.1.2 CPU Memory Validation

The next phase of system power-up is the validation of the CPU memory. First, the system verifies that user memory areas are still valid. A known area of user memory is checked to determine if data was preserved. Next, if a ladder diagram program exists, a checksum is calculated across the _MAIN ladder block. If no ladder diagram program exists, a checksum is calculated across the smallest standalone C program.

When the system is sure that the user memory is preserved, a known area of the bit cache area is checked to determine if the bit cache data was preserved. If this test passes, the Bit Cache memory is left containing its power-up values. (Non-retentive outputs are cleared on a transition from Stop to Run mode.) If the checksum is not valid or the retentive test on the user memory fails, the bit cache memory is assumed to be in error and all areas are cleared. The CPU is now in a cleared state, the same as if a new CPU module were installed. All logic and configuration files must be stored from the programmer to the CPU.

5.11.1.3 System Configuration

After completing its self-test, the CPU performs the system configuration. It first clears all system diagnostic bits in the bit cache memory. This prevents faults that were present before power-down but are no longer present from accidentally remaining as faulted.

5.11.1.4 I/O System Initialization

The operation of input and output defaults for remote I/O modules in an RXi PROFINET network is described in “11.7.5, I/O Defaults Operation.“Power-Down Sequence

System power-down occurs when the CPU detects that incoming DC power has dropped below 18VDC for more than 15ms.

5.11.2 Retention of Data Memory across Power Failure

User memory is not retained across a power failure.

Chapter 6. Program Organization

This chapter provides information about the operation of application programs in a PACSystems CPU.

- Structure of the Application Program
- Controlling Program Execution
- Interrupt-Driven Blocks

6.1 Structure of a PACSystems Application Program

A PACSystems application consists of one block-structured application program. The application program contains all the logic needed to control the operations of the CPU and the modules in the system. Application programs are created using the programming software and transferred to the CPU. Programs are stored in the CPU's non-volatile memory.

During the CPU Sweep (described in Chapter 5), the CPU reads input data from the modules in the system and stores the data in its configured input memory locations. The CPU then executes the entire application program once, using this fresh input data. Executing the application program creates new output data that is placed in the configured output memory locations.

After the application program completes its execution, the CPU writes the output data to modules in the system.

A block-structured program always includes a _MAIN block. Program execution begins with the _MAIN block. Counting the _MAIN block, the program can contain up to 512 blocks.

6.1.1 Blocks

A block is a named section of executable logic that can be downloaded to and run on the target controller. The logic in a block can include functions, function blocks and calls to other blocks.

6.1.2 Functions and Function Blocks

A function is a type of instruction that has no internal storage (instance data). Therefore, it produces the same result for the same set of input values every time it executes.

A function block defines data as a set of inputs and output parameters that can be used as software connections to other blocks and internal variables. It has an algorithm that runs every time the function block is executed. Because a function block has instance data, that is it can store values, it has a defined state.

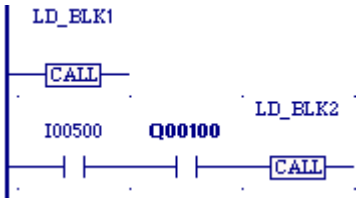
The following types of instructions make up the PACSystems instruction set.

Instruction Type	*Instance Data*	*Examples*
Functions	None	BIT_SEQ, ADD, RANGE
Built-in function blocks	WORD array.	TMR, PID_IND, PID_ISA
Standard function blocks	Structure variable. (See "Instance Data Structures" on page 99.)	TP, TOF, TON

Note: A user defined function block (UDFB) is a block of logic that can be called in your program logic to create multiple instances of the block, allowing you to create a block of logic once and reuse it as if it was a standard function block instruction. For additional information, see pages 93 and 98.

6.1.3 How Blocks Are Called

A block executes when called from the program logic in the _MAIN block or another block. In this example, LD_BLK1 is always called. Conditional logic can be used to control calling a block. For LD_BLK2 to be called, input %I00500 and output %Q00100 must be ON. For details on using the Call function, refer to the Proficy Machine Edition online help.



6.1.4 Nested Calls

The CPU allows nested block calls as long as there is enough execution stack space to support the call. If there is not enough stack space to support a given block call, an "Application Stack Overflow" fault is logged. In these circumstances, the CPU cannot execute the block. Instead, it sets all of the block's Boolean outputs to FALSE, and resumes execution at the point after the block call instruction.

Note: To halt the CPU when there is not enough stack space to execute a block, there are two choices. The best method is to add logic to detect the occurrence of any User Application Fault by testing the diagnostic bit %SA38, and then call SVC_REQ 13 to halt the CPU. An alternative method is to add logic that tests for a negative OK value coming out of the block and then call SVC_REQ 13 to halt the CPU.

A call depth of eight levels or more can be expected, except in rare cases where several of the called blocks have very large numbers of parameters. The actual call depth achieved depends on several factors, including the amount of data (non-Boolean) flow used in the blocks, the particular functions called by the blocks, and the number and types of parameters defined for the blocks. If blocks use less than the maximum amount of stack resources, more than eight nested calls may be possible. The call level nesting counts the _MAIN block as level 1.

6.1.5 Types of Blocks

PACSystems supports four types of blocks.

Block Type	Local Data	Programming Languages	Size Limit	Parameters
Block	Has its own local data	LD FBD ST	128 KB	0 inputs 1 output
Parameterized Block	Inherits local data from caller	LD FBD ST	128 KB	63 inputs 64 outputs
User Defined Function Block (UDFB)	Has its own local data	LD FBD ST	128 KB	63 inputs 64 outputs Unlimited internal member variables
External Block	Inherits local data from caller	C	user memory size limit (10 MB)	63 inputs 64 outputs

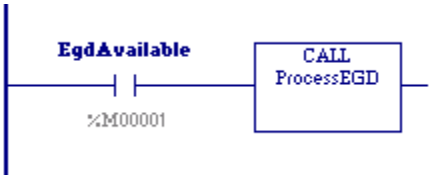
All PACSystems block types automatically provide an OK output parameter. The name used to reference the OK parameter within a block is Y0. Logic within the block can read and write the Y0 parameter. When a block is called, its Y0 parameter is automatically initialized to TRUE. This will result in a positive power flow out of the block call instruction when the block completes execution, unless Y0 is set to FALSE within the logic of the block.

For all block types, the maximum number of input parameters is one less than the maximum number of output parameters. This is because the EN input to the block call is not considered to be an input parameter to the block. It is used in LD language to determine whether or not to call the block, but is not passed into the block if the block is called.

6.1.5.1 Program Blocks

Any block can be a program block. The _MAIN block is automatically declared when you create a block-structured program. When you declare any other block, you must assign it a unique block name. A block is automatically configured with no input parameters and one output parameter (OK).

When a block-structured program is executed, the _MAIN block is automatically executed. Other blocks execute when called from the program logic in the _MAIN block, another block, or itself. In the following example, if %M00001 is ON, the block named ProcessEGD will be executed:



Program Blocks and Local Data

Program blocks support the use of %P global data. In addition, each block, except _MAIN, has its own %L local data. Blocks do not inherit %L local data from their callers.

Using Parameters with a Program Block

Every block is automatically defined to have one formal ‘power flow’ (or OK) output parameter, named Y0. Y0 is a BOOL parameter of LENGTH 1, passed by initial-value result. It indicates successful execution of the block. It can be read and written to by the logic within the block.

6.1.5.2 Parameterized Blocks

Any block except _MAIN can be a parameterized block. When you declare a parameterized block, you must assign it a unique block name. A parameterized block can be configured with up to 63 input and 64 output parameters.

A parameterized block executes when called from the program logic in the _MAIN block, another block, or itself. In the following example, if %I00001 is set, the parameterized block named LOAD_41 will be executed.

Parameterized Blocks and Local Data

Parameterized blocks support the use of %P global data. Parameterized blocks do not have their own %L data, but instead inherit the %L data of their calling blocks. Parameterized blocks also inherit the FST_EXE system reference and “time stamp” data that is used to update timer functions from their calling blocks. If %L references are used within a parameterized block and the block is called by _MAIN, %L references will be inherited from the %P references wherever encountered in the parameterized block (for example, %L0005 = %P0005).

Note: It is possible, by using Online Editing in the programming software to cause a parameterized block to use %L higher than allowed because of the way it inherits data. Using a word-for-word change to restore this reference to a valid address does not correct the block because the variable still exists in the variable list. Deleting the variable from the variable list does not cause an update to the CPU, so the parameterized block still sees the reference out of range fault. To correct this condition, you must remove the unused variables from the variable list after deleting them from the logic.

Using Parameters with a Parameterized Block

A parameterized block may be defined to have between 0 and 63 formal input parameters, and between 1 and 64 formal output parameters. A 'power-flow out' (or OK) parameter, named Y0, is automatically defined for every parameterized block. It is a BOOL parameter of LENGTH 1, and indicates the successful execution of the parameterized block. It can be read and written to by the parameterized block's logic.

The following table lists the TYPEs, LENGTHs, and parameter-passing mechanisms allowed for parameterized block parameters. (For definitions of the parameter passing types, see "Parameter Passing Mechanisms" on page 105.)

Type	Length	Default Parameter Passing Mechanism
BOOL	1 to 256	INPUTS: by reference
		OUTPUTS: by value result; except Y0, which is by initial-value result
BYTE	1 to 1024	INPUTS: by reference
		OUTPUTS: by reference
INT, UINT, and WORD	1 to 512	INPUTS: by reference
		OUTPUTS: by reference
DINT, REAL, and DWORD	1 to 256	INPUTS: by reference
		OUTPUTS: by reference
LREAL	1 to 128	INPUTS: by reference
		OUTPUTS: by reference
function block*	1	INPUTS: by reference
		OUTPUTS: not allowed
UDFB*	1	INPUTS: by reference
		OUTPUTS: not allowed
User Defined Type (UDT)	1 to 1024	INPUTS: by reference
		OUTPUTS: not allowed

* A maximum of 16 input parameters can be of type function block or UDFB.

The PACSystems default parameter passing mechanisms correspond to the way that parameterized subroutine block (PSB) parameters are passed on 90-70 controllers. The parameter passing mechanisms of formal parameters cannot be changed from their default values.

Arguments or “actual parameters” are passed into a parameterized block when a parameterized block call is executed. In general, arguments to formal parameters may come from any memory type, may be data flow, and may be constants (when the formal parameter’s LENGTH is 1). The following list contains the restrictions on arguments relative to this general rule:

- %S memory addresses cannot be used as arguments to any output parameter. This is because user logic is not allowed to write to %S memory.
- Indirect references used as arguments are resolved immediately before the parameterized block is called, and the corresponding direct reference is passed into the block. For example, where %R1 contains the value 10 and @R1 is used as an argument to a call, immediately before calling the block, @R1 is resolved to be %R10, and %R10 is passed in as the argument to the block. During execution of the block, the argument remains as %R10, regardless of whether the value in %R1 changes.

In general, formal parameters within a parameterized block may be used with any instruction or with any block call, as long as their TYPE and LENGTH are compatible with what the instruction, function, or block call requires. The following list contains the restrictions on formal parameters relative to this general rule:

- Formal parameters cannot be used on legacy transitional contacts or coils, or on FAULT, NOFLT, HIALM, or LOALM contacts. However, formal parameters can be used on IEC transitional contacts and coils.
- Formal BOOL input parameters cannot be used on coils or as output arguments to a function or to a block call.
- Formal parameters cannot be used with the DO I/O function.
- Formal parameters cannot be used with indirect referencing.

6.1.5.3 User Defined Function Blocks

Users can define their own blocks, which have parameters and instance data, instead of being limited to the standard and built-in function blocks provided in the PACSystems instruction set. In many cases, the use of this feature results in a reduction in total program size.

A *member variable* is not passed into or out of a UDFB as a parameter. A member variable is used only within the logic of a function block.

Once defined, multiple instances of a UDFB can be created by calling it within the program logic. Each instance has its own unique copy of the function block's *instance data*, which consists of the function block's internal member variables and all of its input and output parameters except those that are passed by reference. When a UDFB is called on a given instance, the UDFB's logic operates on that instance's copy of the instance data. The values of the instance data persist from one execution of the UDFB to the next.

A UDFB cannot be triggered by an interrupt.

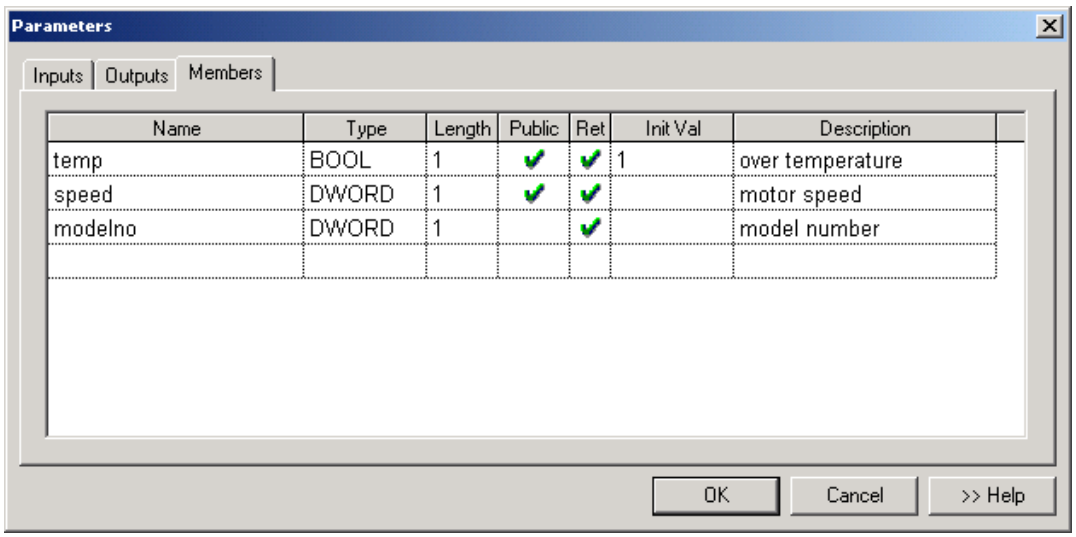
UDFB logic is created using FBD, LD or ST. UDFB logic can make calls to all the other types of PACSystems blocks (blocks, parameterized blocks, external blocks and other UDFBs). Blocks, parameterized blocks, and other UDFBs can make calls to UDFBs.

Unless otherwise stated, the PACSystems implementation of UDFBs meets the IEC 61131-3 requirements for user defined function blocks.

Defining a UDFB

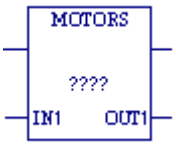
To create a UDFB in the programming software, create an LD, FBD or ST block in the Program Blocks folder. In the Properties for the block, select Function Block.

To define instance data for a UDFB, select Parameters in the block's properties. Input and output parameters are defined in the same way as for parameterized blocks. In the following example, three internal member variables are defined: ***temp, speed,*** and ***modelno.***

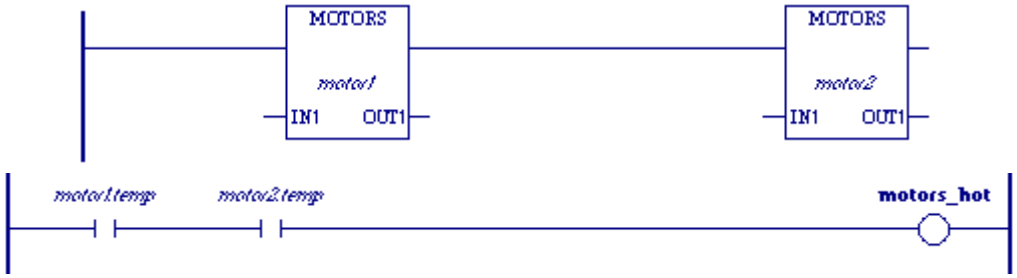


Creating UDFB Instances

You create an instance of a UDFB by calling it in your logic and assigning an instance name in the function properties.

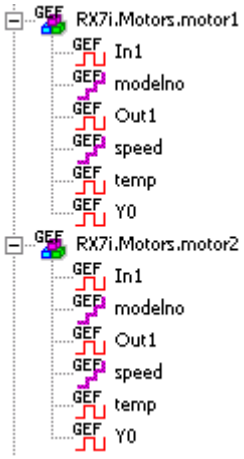


In the following LD example, the first rung creates two instances of the UDFB, Motors. The instance variables associated with the instances are motors.motor1 and motors.motor2. The second rung uses the two instances of the internal variable ***temp*** in logic.



Instance Data Structures

A variable with the format *function_block_name.instance_name* is automatically created for each instance of a UDFB. The instance data makes up a single composite variable that is of a structure type. The example to the right shows the variable structures associated with two instances of the UDFB named Motors. Each instance variable has elements corresponding to parameters *In1, Out1*, and *Y0*, and internal variables *modelno, speed,* and *temp*.



Instances are created as symbolic variables, never as mapped variables. This ensures that instance data is only referenced by the instance name and not by a memory address, which means that no aliases can be created for the UDFB data elements. The indirect reference operator cannot be used on an instance variable because indirect references are not permitted on symbolic variables.

UDFBs and Scope

Unlike a parameterized subroutine, a UDFB has its own %L memory.

By default, internal variables of a UDFB have local scope, making them visible only to the logic inside the UDFB. They cannot be read or written by any external logic or by the hardware configuration. An internal variable can be made visible outside the UDFB by changing its scope to global. Logic outside the UDFB can read but cannot write to internal variables whose scope is global.

Note: If you give internal variables global scope, your application will not conform to IEC requirements.

Using Parameters with UDFBs

UDFBs support up to 63 inputs and up to 64 outputs.

Each UDFB has a predefined Boolean output parameter, Y0, which the CPU sets to true upon each invocation of the block. Y0 can be controlled by logic within the block and provides the output status of the block.

The following table lists the TYPEs, LENGTHs, and parameter-passing mechanisms allowed for UDFB parameters. For additional information on parameter passing, see “Parameter Passing Mechanisms” on page 105.

Type	Length	Parameter Passing Mechanism	Retentiveness of Instance Data for Parameters
BOOL	1 to 256	INPUTS: by reference, constant reference, value, or value result. (Default: value)	Not Applicable if passed by reference, since not stored in instance data. Can be retentive (default) or nonretentive for value or value result.
		OUTPUTS: by result; except Y0, which is by initial-value result	Retentive (default) or Nonretentive
BYTE	1 to 1024	INPUTS: by reference, constant reference, value, or value result. (Default: value)	Retentive for value or value result. Not applicable for reference
		OUTPUTS: by result	
INT, UINT, and WORD	1 to 512	INPUTS: by reference, constant reference, value, or value result. (Default: value)	Retentive for value or value result. Not applicable for reference
		OUTPUTS: by result	
DINT, REAL, and DWORD	1 to 256	INPUTS: by reference, constant reference, value, or value result. (Default: value)	Retentive for value or value result. Not applicable for reference
		OUTPUTS: by result	
LREAL	1 to 128	INPUTS: by reference, constant reference, value, or value result. (Default: value)	Retentive for value or value result. Not applicable for reference
		OUTPUTS: by result	
Function block (standard or PACMotion)	1	INPUTS: by reference, constant reference, (Default: reference)	Not applicable since passed by reference
		OUTPUTS: by result	
UDFB*	1	INPUTS: by reference, constant reference, friend	Not applicable since passed by reference
		OUTPUTS: not allowed	
UDT	1 to 1024	INPUTS: by reference, constant reference	Not applicable since passed by reference
		OUTPUTS: not allowed	

* A maximum of 16 input parameters can be of type UDFB.

If an input parameter is passed by reference or by value result, it requires an argument. All other parameters of a UDFB are optional. That is, they do not have to be given arguments on each instance of the UDFB. If no argument is given for an optional parameter, the variable element associated with the parameter retains the value it previously had.

UDFB outputs cannot be passed as arguments to input parameters that are passed by reference or passed by value result. This restriction prevents modification of a UDFB output.

Using Internal Member Variables with UDFBs

A UDFB can have any number of internal member variables. Internal variables' values are not passed through the input and output parameters. An internal variable cannot have the same name as a parameter of the UDFB it is defined in.

An internal variable can be:

- Any basic type supported by PACSystems (BOOL, INT, UINT, DINT, REAL, LREAL, BYTE, WORD, and DWORD).
- A UDFB type. Such member variables are known as nested instances. For example, the function block "Motor" can have an internal variable of type "Valve," where Valve is a UDFB type. Note that defining a member variable as a UDFB type does not create an instance.

  A nested instance cannot be of the same type as the UDFB being defined because this would set up an infinitely recursive definition. Nor can any level of a nested instance be of the same type as the parent UDFB being defined. For example, the UDFB "Motor" cannot have an internal variable of type "Valve," if the Valve UDFB contains an internal variable of type "Motor."
- A UDT. A structured, user-defined data type consisting of elements of other selected data types.
- A one-dimensional array.

Internal variables of TYPE BOOL can be retentive (default) or nonretentive. All other TYPEs must be retentive.

Member variables corresponding to a UDFB's input parameters cannot be read or written outside of the UDFB. (This is more restrictive than the IEC 61131-3 requirements for user defined function blocks.) Member variables corresponding to the UDFB's output parameters can be read but not written outside the UDFB.

Internal member variables that have basic types may be given initial values. The same initial values apply to all instances of a UDFB. If an initial value isn't given, the internal member variable is set to zero when the application transitions to RUN mode for the first time.

An internal member variable that is a nested instance has initial values as specified by its UDFB type definition.

Initial values are not stored during a RUN mode store. They will not take effect until a Stop mode store is performed.

UDFB Logic

An instance of a BOOL parameter or internal variable can be forced ON or OFF, or used with transition-detecting instructions. The exception to this is that BOOL input parameters passed by reference cannot be forced or used with the Series 90-70 legacy transition-detecting instructions (POSCOIL, NEGCOIL, POSCON and NEGCON) because their values are not stored in instance data.

All input parameters to a UDFB, and their corresponding instance data elements, can be read by their UDFB's logic.

Input parameters that are passed by reference or passed by value result to a UDFB can be written to by their UDFB's logic. Input parameters passed by value ***cannot*** be written to by their UDFB logic. Note that the restriction on writing to input parameters passed by value does not apply to other types of blocks.

All UDFB output parameters can be both read and written to by their logic.

UDFB Operation with Other Blocks

A UDFB instance that is of global scope can be invoked by another UDFB's logic or any other block's logic.

A UDFB instance that is passed (by reference) as an argument to a UDFB can be invoked by the UDFB's logic.

A UDFB instance that is passed (by reference) as an argument to a parameterized block can be invoked by the parameterized block's logic.

The output parameters, and their corresponding instance data elements, of a UDFB instance that is passed as an argument can be read but not modified by the receiving block's logic. The input parameters of a UDFB instance that is passed as an argument cannot be read or modified by the receiving block's logic. The internal variables of a UDFB instance that is passed as an argument cannot be modified by the receiving block's logic. They can be read if their scope is global, but not if their scope is local.

6.1.5.4 External Blocks

External blocks are developed using external development tools as well as the C Programmer's Toolkit for PACSystems. Refer to the *C Programmer's Toolkit for PACSystems User's Manual,* GFK-2259 for detailed information regarding external blocks.

Any block except _MAIN can be an external block. When you declare an external block, you must assign it a unique block name. It can be configured with up to 63 input parameters and 64 output parameters.

An external block executes when called from the program logic in the _MAIN block or from the logic in another block, parameterized block, or UDFB. External blocks themselves cannot call any other block. In the following example, if %I00001 is set, the external block named EXT_11 is executed.

Note: Unlike other block types, external blocks cannot call other blocks.

External Blocks and Local Data

External blocks support the use of %P global data. External blocks do not have their own %L data, but instead inherit the %L data of their calling blocks. They also inherit the FST_EXE system reference and the "time stamp" data that is used to update timer function blocks from their calling blocks. If %L references are used within an external block and the block is called by _MAIN, %L references will be inherited from the %P references wherever encountered in the external block (for example, %L0005 = %P0005).

Initialization of C Variables

When an external block is stored to the CPU, a copy of the initial values for its global and static variables is saved. However, if static variables are declared without an initial value, the initial value is undefined and must be initialized by the C application. (Refer to “Global Variable Initialization” and “Static Variable” in the *C Programmer’s Toolkit for PACSystems,* GFK-2259). The saved initial values are used to re-initialize the block’s global and static variables whenever the CPU transitions from Stop to Run.

Using Parameters With an External Block

An external block may be defined to have between zero and 63 formal input parameters and between one and 64 formal output parameters. A ‘power-flow out’ (or OK) parameter, named Y0, is automatically defined for every external block. Y0 is a BOOL parameter of LENGTH 1, and indicates the successful execution of the block. It can be read and written to by the external block’s logic.

The following table gives the TYPEs, LENGTHs, and parameter-passing mechanisms allowed for external block parameters.

Type	*Length*	*Default Parameter Passing Mechanism*
BOOL	1 to 256	INPUTS: by reference
		OUTPUTS: by reference; except Y0, which is by initial-value result
BYTE	1 to 1024	INPUTS: by reference
		OUTPUTS: by reference
INT, UINT, and WORD	1 to 512	INPUTS: by reference
		OUTPUTS: by reference
DINT, REAL, and DWORD	1 to 256	INPUTS: by reference
		OUTPUTS: by reference
LREAL	1 to 128	INPUTS: by reference
		OUTPUTS: by reference
UDT*	1 to 128	INPUTS: by reference
		OUTPUTS: not allowed

* To use a UDT, you must include the UDT definition as a C structure in the external block. For details, see "Using a UDT as a C block input parameter data type" in the online help.

The PACSystems default parameter passing mechanisms correspond to the way that external block parameters are passed on 90-70 controllers. The parameter passing mechanisms of formal parameters cannot be changed from their default values.

You must define a name for each formal input and output parameter.

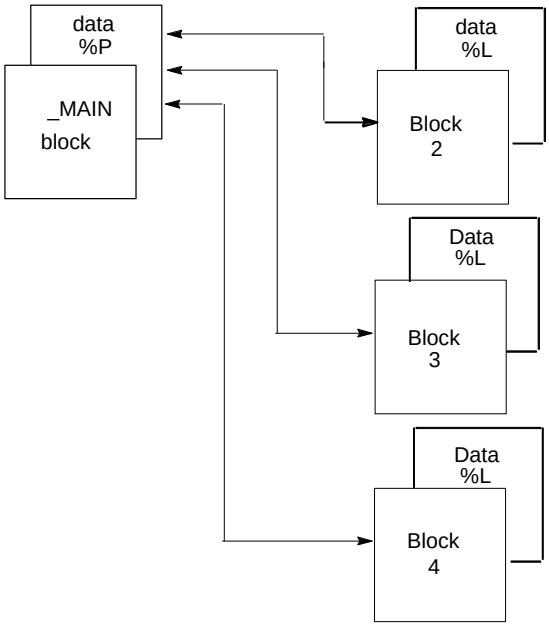
Arguments, or “actual parameters”, are passed into an external block when an external block call is executed.

Arguments may be any valid reference address including an indirect reference, may be flow, or may be a constant if the corresponding parameter’s LENGTH is 1.

6.1.6 Local Data

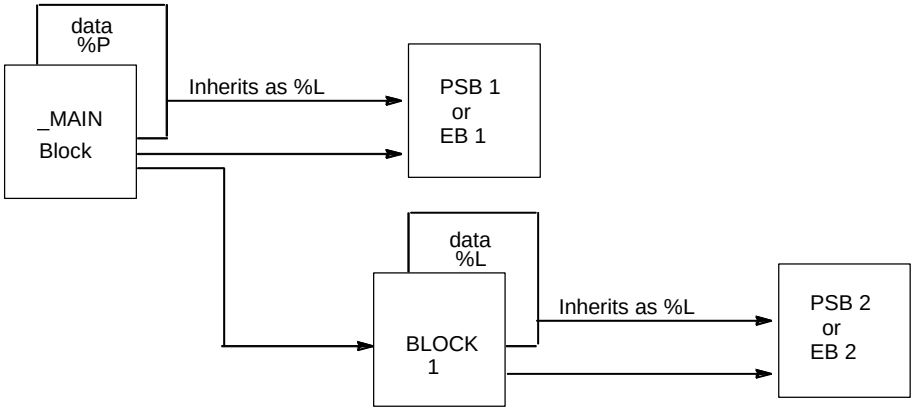
Each block or UDFB in a block-structured program has an associated local data block. _MAIN's data block memory is referenced by %P; all other data block memories are referenced by %L.

The size of the data block is dependent on the highest reference in its block for %L and in all blocks for %P.



All blocks within the program can use data associated with the _MAIN block (%P). Blocks and UDFBs can use their own %L data as well as the %P data that is available to all blocks. The _MAIN block cannot use %L.

External blocks and parameterized blocks can use the Local Data (%L) of their calling block as well as the %P data of the _MAIN block. If a parameterized block or external block is called by MAIN, all %L references in the parameterized block or external block will actually be references to corresponding %P references (for example, %L0005 = %P0005). In addition to inheriting the Local Data of their calling blocks, parameterized blocks and external blocks inherit the FST_EXE status of their calling blocks.



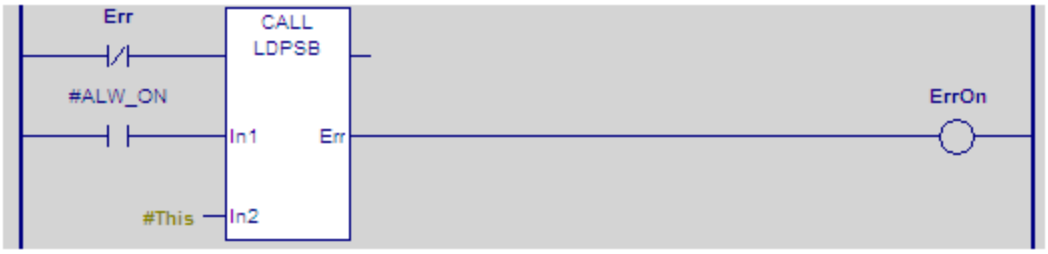
6.1.7 Parameter Passing Mechanisms

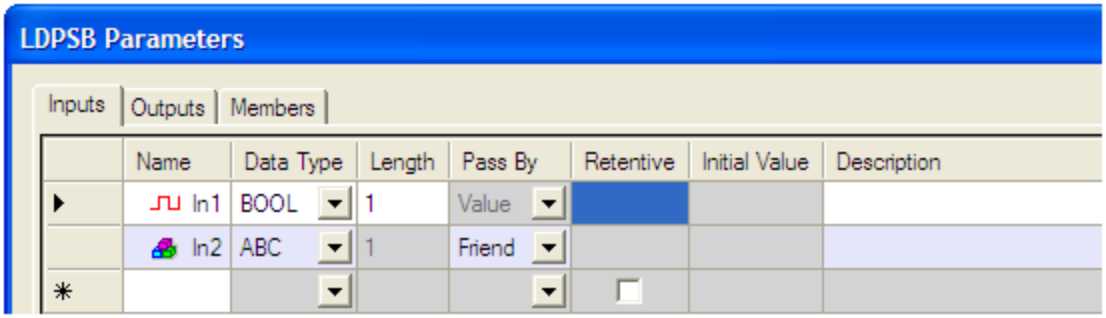
All blocks (except _MAIN) have at least one parameter and thus are affected by parameter passing mechanisms. A “parameter passing mechanism” describes the way that data is passed from an argument in a calling block to a parameter in the called block, and from the parameter in the called block back to the argument in the calling block.

PACSystems supports the following parameter-passing mechanisms: *pass by reference, pass by constant reference, pass by value, pass by value result, pass by result* and *pass by initial-value result.* An additional type, *pass by friend,* is available when the input Data Type is a UDFB. A parameter is defined by its TYPE, LENGTH, and parameter passing mechanism.

- When a parameter is passed by **reference**, the address of its argument is passed into the function block instance or parameterized block. All logic within the called block that reads or writes to the parameter directly reads or writes to the actual argument.
- When a parameter is passed by **constant reference**, the CPU passes a reference address pointer, symbolic variable pointer, or I/O variable pointer into the function block instance or parameterized block. The instance or block can only read the reference address or variable.
- When a parameter is passed by **friend** (UDFB inputs only), the CPU passes a UDFB instance variable pointer into the function block instance or parameterized block. The instance or block can write to any output or member, whether public or private, of the UDFB instance variable passed as a friend.

  Tip: In the logic of a UDFB, when you want to pass the UDFB as a friend, assign the pseudo-variable "#This" to the input that expects an instance variable of that UDFB type. In the following example, the In2 input of the LDPSB parameterized block expects a UDFB instance variable friend of the ABC data type. Inside the logic of ABC, assign "#This" to In2 in the call to LDPSB.





- When a parameter is passed by **value (UDFB inputs only),** the value of its argument is copied into a local stack memory associated with the called block. All logic within the called block that reads or writes to the parameter is reading or writing to this stack memory. Thus no changes are ever made to the actual argument.
- When a parameter is passed by **value result (UDFB inputs only),** the value of its argument is copied into a local stack memory associated with the called block, and the address of its argument is saved. All logic within the called block that reads or writes to the parameter is reading or writing to this stack memory. When the called block completes its execution, the value in the stack memory is copied back to the actual argument's address. Thus no changes are made to the actual argument while the called block is executing, but when it completes execution, the actual argument is updated.

6.1.8 Languages

The RXi Controller supports the following IEC 61131-3 programming languages:

- Ladder Diagram (LD)
- Structured Text
- Function Block Diagram

6.1.9 RXi Controller Instruction Set

The following instructions and standard library functions make up the RXi CPU's *instruction set.* For detailed information on using these programming elements, refer to the Proficy Machine Edition online help.

Bit Operation
AND_WORD
AND_DWORD
OR_WORD
OR_DWORD
XOR_WORD
XOR_DWORD
NOT_WORD
NOT_DWORD
MCMP_WORD
MCMP_DWORD
SHL_WORD
SHL_DWORD
SHR_WORD
SHR_DWORD
ROL_WORD
ROL_DWORD
ROR_WORD
ROR_DWORD
BTST_WORD
BTST_DWORD
BSET_WORD
BSET_DWORD
BCLR_WORD
BCLR_DWORD
BPOS_WORD
BPOS_DWORD
Relational
CMP_INT
CMP_DINT
CMP_REAL
CMP_LREAL
CMP_UINT
EQ_DATA
EQ_DINT
EQ_INT
EQ_LREAL
EQ_REAL
EQ_UINT
NE_INT
NE_DINT
NE_UINT
NE_REAL
NE_LREAL

GT_INT
GT_DINT
GT_REAL
GT_LREAL
GT_UINT
GE_INT
GE_DINT
GE_REAL
GE_LREAL
GE_UINT
LT_INT
LT_DINT
LT_REAL
LT_LREAL
LT_UINT
LE_INT
LE_DINT
LE_UINT
LE_REAL
LE_LREAL
Conversion
BCD-4 to INT
DINT to INT
UINT to INT
BCD-8 to DINT
INT to DINT
UINT to DINT
INT to UINT
DINT to UINT
BCD-4 to UINT
INT to BCD-4
UINT to BCD-4
DINT to BCD-8
REAL_TO_INT
REAL_TO_UINT
REAL_TO_LREAL
REAL_TO_DINT
INT_TO_REAL
UINT_TO_REAL
DINT_TO_REAL
DINT_TO_LREAL
REAL_TRUN_INT
REAL_TRUN_DINT
DEG_TO_RAD_REAL

DEG_TO_RAD_LREAL
RAD_TO_DEG_REAL
RAD_TO_DEG_LREAL
BCD-4 to REAL
BCD-8 to REAL
LREAL_TO_DINT
LREAL_TO_REAL
Data Move
BLKCLR
BITSEQ
MOVE_BIT
MOVE_DINT
MOVE_INT
MOVE_UINT
MOVE_WORD
MOVE_DWORD
MOVE_REAL
MOVE_LREAL
MOVE_DATA
MOVE_DATA_EX
MOVE_TO_FLAT
MOVE_FROM_FLAT
BLKMOV_WORD
BLKMOV_DWORD
BLKMOV_DINT
BLKMOV_INT
BLKMOV_REAL
BLKMOV_UINT
DAT_INIT_ASCII
DAT_INIT_COMM
DAT_INIT_DLAN
DAT_INIT_DINT
DAT_INIT_DWORD
DAT_INIT_INT
DAT_INIT_REAL
DAT_INIT_LREAL
DAT_INIT_WORD
DAT_INIT_UINT
SWAP_WORD
SWAP_DWORD
SHFR_BIT
SHFR_WORD
SHFR_DWORD

Data Table
SORT_INT
SORT_UINT
SORT_WORD
TBLRD_INT
TBLRD_DINT
TBLWRT_INT
TBLWRT_DINT
FIFORD_INT
FIFORD_DINT
FIFOWRT_INT
FIFOWRT_DINT
LIFORD_INT
LIFORD_DINT
LIFOWRT_INT
LIFOWRT_DINT
LIFOWRT_DWORD
Array
ARRAY_MOVE_BIT
ARRAY_MOVE_BYTE
ARRAY_MOVE_WORD
ARRAY_MOVE_DWORD
ARRAY_MOVE_DINT
ARRAY_MOVE_INT
ARRAY_MOVE_UINT
SRCH_BYTE
SRCH_WORD
SRCH_DWORD
ARRAY_RANGE_ WORD
ARRAY_RANGE_ DWORD
ARRAY_RANGE_DINT
ARRAY_RANGE_INT
ARRAY_RANGE_UINT

Math
ADD_INT
ADD_DINT
ADD_REAL
ADD_LREAL
ADD_UINT
SUB_INT
SUB_DINT
SUB_REAL
SUB_LREAL
MUL_INT
MUL_DINT
MUL_REAL
MUL_LREAL
MUL_MIXED
MUL_UINT
DIV_INT
DIV_DINT
DIV_REAL
DIV_LREAL
DIV_MIXED
MOD_INT
MOD_DINT
MOD_UINT
ABS_INT
ABS_DINT
ABS_REAL
ABS_LREAL
SCALE_INT
SCALE_DINT
SCALE_UINT
SQRT_INT
SQRT_DINT
SQRT_REAL
SQRT_LREAL

Trigonometric
SIN_REAL
SIN_LREAL
COS_REAL
COS_LREAL
TAN_REAL
TAN_LREAL
ASIN_REAL
ASIN_LREAL
ACOS_REAL
ACOS_LREAL
ATAN_REAL
ATAN_LREAL
Logarithmic
LOG_REAL
LOG_LREAL
LN_REAL
LN_LREAL
EXPT_REAL
EXPT_LREAL
EXP_REAL
EXP_LREAL
PID
PIDISA
PIDIND
Range
RANGE_INT
RANGE_DINT
RANGE_DWORD

Timers
ONDTR
OFDT
TMR
TOF
TON
TP
Counters
UPCTR
DNCTR
Control
JUMPN
FOR/NEXT
MCRN/ENDMCRN Combined
DOIO
DOIO with ALT
DRUM_SEQ
SCAN_SET_IO
SUS_IO
COMM_REQ
CALL/RETURN (C Block)
CALL/RETURN (LD)
CALL/RETURN (PSB)
Service Requests (SVC_REQ)
1,Change/read constant sweep timer
2,Read window modes and time values
3,Change controller communications window mode and timer value
4,Change backplane communications window mode and timer value

5,Change background task window mode and timer value
6,Change/read number of words to checksum
7,Read or change the time-of-day clock
8,Reset watchdog timer
9,Read sweep time from beginning of sweep - milliseconds
10,Read target name
11,Read controller ID
12,Read controller run state
13,Shut down (stop) controller
14,Clear controller or I/O fault tables
15,Read last-logged fault table entry
16,Read elapsed time clock - microseconds
18,Read I/O override status
19,Set run enable/disable
20,Read fault tables
21,User-defined fault logging
22,Mask/unmask timed interrupts
23,Read master checksum
50,Read elapsed time clock – nanoseconds
51,Read sweep time from beginning of sweep - nanoseconds

6.2 Controlling Program Execution

There are many ways in which program execution can be controlled to meet the system's timing requirements. The PACSystems CPU instruction set contains several powerful control functions that can be included in an application program to limit or change the way the CPU executes the program and scans I/O. For details on using these functions, refer to the Proficy Machine Edition online help.

The following is a partial list of commonly used methods:

- The Jump (JUMPN) function can be used to cause program execution to move either forward or backward in the logic. When a JUMPN function is active, the coils in the part of the program that is skipped are left in their previous states (not executed with negative power flow, as they are with a Master Control Relay). Jumps cannot span blocks.
- The nested Master Control Relay (MCRN) function can be used to execute a portion of the program logic with negative power flow. Logic is executed in a forward direction and coils in that part of the program are executed with negative power flow. Master Control Relay functions can be nested to 255 levels deep.
- The Suspend I/O function can be used to stop both the input scan and output scan for one sweep. I/O can be updated, as necessary, during the logic execution through the use of DO I/O instructions.
- The Service Request function can be used to suspend or change the time allotted to the window portions of the sweep.
- Program logic can be structured so that blocks are called more or less frequently, depending on their importance and on timing constraints. The CALL function can be used to cause program execution to go to a specific block. Conditional logic placed before the Call function controls the circumstances under which the CPU executes the block logic. After the block execution is finished, program execution resumes at the point in the logic directly after the CALL instruction.

6.3 Interrupt-Driven Blocks

Timed interrupts can be used to start a block's execution. These interrupts are generated by the CPU based on a user-specified time interval with an initial delay (if specified) applied on Stop-to-Run transition of the CPU.

Caution

Interrupt-driven block execution can interrupt the execution of non-interrupt-driven logic. Unexpected results may occur if the interrupting logic and interrupted logic access the same data. If necessary, Service Request #17can be used to temporarily mask Timed Interrupt-driven logic from executing when shared data is being accessed.

6.3.1 Interrupt Handling

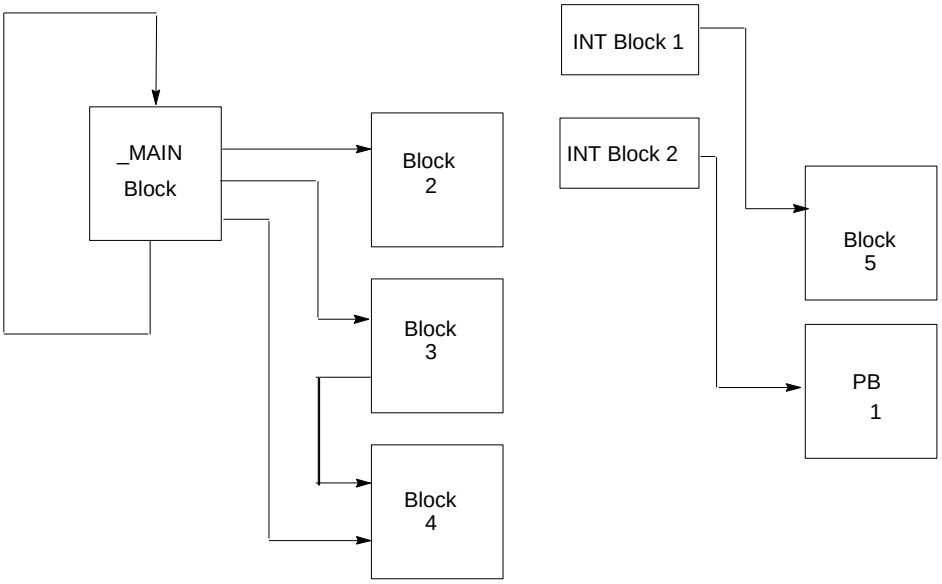
A Timed interrupt can be associated with any block except _MAIN, as long as the block has no parameters other than an OK output. After an interrupt has been associated with a block, that block executes each time the interrupt trigger occurs. A given block can have multiple timed interrupt triggers associated with it. It is executed each time any one of its associated interrupts triggers. For details on how interrupt blocks are prioritized, refer to "Interrupt Block Scheduling" on page 111.

If a parameterized block or external block is triggered by an interrupt, it inherits %P data as its %L local data. For example, a %L00005 reference in the parameterized block or C block actually references %P00005.

Note: Timer function blocks do not accumulate time if used in a block that is executed as a result of an interrupt.

Blocks that are triggered by interrupts can make calls to other blocks. The application stack used during interrupt-driven execution is different from the stack used during normal block-structured program execution. In particular, the nested call limit is different from the limit described for calls from the _MAIN block. If a call results in insufficient stack space to complete the call, the CPU logs an ”Application Stack Overflow” fault.

Note: We strongly recommend that interrupt-driven blocks not be called from the _MAIN block or other non-interrupt driven blocks because the interrupt and non-interrupt driven blocks could be reading and writing the same global memories at indeterminate times relative to each other. In the example below INT1, INT2, BLOCK5, and PB1 should not be called from _MAIN, BLOCK2, BLOCK3, or BLOCK4.



6.3.2 Configuring Timed Interrupts

A block can be configured to execute on a specified time interval with an initial delay (if specified) applied on a Stop-to-Run transition of the CPU.

To configure a timed interrupt block, specify the following parameters in the scheduling properties for the block:

Time Base	The smallest unit of time that you can specify for Interval and Delay. The time base can be 1.0 second, 0.10 second, or 0.01 second, or 0.001 second.
Interval	Specifies how frequently the block executes in multiples of the time base.
Delay	(Optional) Specifies an additional delay for the first execution of the block in multiples of the time base.

The first execution of a Timed Interrupt block will occur at ((delay * time base) + (interval * time base)) after the CPU is placed in Run mode.

6.3.3 Interrupt Block Scheduling

You can select one of two types of interrupt block scheduling at the target level:

- **Normal block scheduling** allows you to associate a maximum of 16 Timed Interrupts. With normal block scheduling, all interrupt-triggered blocks have equal priority. This is the default scheduling mode.
- **Preemptive block scheduling** allows you to associate a maximum of 32 interrupt triggers. With preemptive block scheduling, each trigger can be assigned a relative priority.

6.3.3.1 Normal Block Scheduling

Interrupt-driven logic has the highest priority of any user logic in the system. The execution of a block triggered from an interrupt preempts the execution of the normal CPU sweep activities. Execution of the normal CPU sweep activities is resumed after the interrupt-driven block execution completes.

If the CPU receives one or more interrupts while executing an interrupt block, it places the incoming interrupts into the queue while it finishes executing the current interrupt block. If an interrupt driven block is already in the queue, additional interrupts that occur for this block are ignored.

6.3.3.2 Preemptive Block Scheduling

Preemptive scheduling allows you to assign a priority to each interrupt trigger. The priority values range from 1 to 16, with 1 being the highest. A single block can have multiple interrupts with different priorities or the same priorities.

An incoming interrupt is handled according to its priority compared to that of the currently executing block as follows:

- If an incoming interrupt has a higher priority than the interrupt associated with the block that is currently executing, the currently executing block is stopped and put in the interrupt queue. The block associated with the incoming interrupt begins executing.
- If an incoming interrupt has the same priority as the interrupt trigger associated with the block that is currently executing, that block continues to execute and the incoming interrupt is placed in the queue.
- If an incoming interrupt has a lower priority than the interrupt associated with the block that is currently executing, the incoming interrupt is placed in the queue.

When the CPU completes the execution of an interrupt block, the block associated with the interrupt trigger that has the highest priority in the queue begins execution — or resumes execution if the block's execution was preempted by another interrupt block and was placed in the queue.

If multiple blocks in the queue have the same interrupt priority, their execution order is not deterministic.

Note: Certain functions, such as DOIO and some SVC_REQs may cause a block to yield to another queued block that has the same priority.

Chapter 7. CPU Program Data

This chapter describes the types of data that can be used in an application program, and explains how that data is stored in the PACSystems CPU's memory.

- Variables
- Reference Memory
- User Reference Size and Default
- Transitions and Overrides
- Retentiveness of Logic and Data
- Data Scope
- System Status References
- How Program Functions Handle Numerical Data
- User Defined Types (UDTs)
- Word-for-Word Changes
- Operands for Instructions

7.1 Variables

A variable is a named storage space for data values. It represents a memory location in the target PACSystems CPU.

A variable can be mapped to a reference address (for example, %R00001). If you do not map a variable to a specific reference address, it is considered a *symbolic variable*. The programming software handles the mapping for symbolic variables in a special portion of PACSystems user space memory.

A variable's data type determines the kinds of values it can store. For example, variables with a UINT data type store unsigned whole numbers with no fractional part. Data types are described in "How Program Functions Handle Numerical Data" on page 128.

In the programming software, all variables in a project are displayed in the Variables tab of the Navigator. You create, edit, and delete variables in the Variables tab. Some variables are also created automatically by certain components (such as TIMER variables when you add a Timer instruction to ladder logic). The data type and other properties of a variable, such as reference address are configured in the Inspector.

For more information about *system variables,* which are created when you create a target in the programming software, refer to page 125.

7.1.1 Mapped Variables

Mapped (manually located) variables are assigned a specific reference address. For details on the types of reference memory and their uses, refer to page 119.

7.1.2 Symbolic Variables

Symbolic variables are variables for which you do not specify a reference address (similar to a variable in a typical high-level language). Except as noted in this section, you can use these in the same ways that you use mapped variables.

In the programming software, a symbolic variable is displayed with a blank address. You can change a mapped variable to a symbolic variable by removing the reference address from the variable's properties. Similarly, you can change a symbolic variable into a mapped variable by specifying a reference address for the variable in its properties.

The memory required to support symbolic variables counts against user space. The amount of space reserved for these variables is configured on the Memory tab in the CPU hardware configuration.

7.1.2.1 Restrictions on the Use of Symbolic Variables

- Symbolic variables cannot be used with indirect references (for example, @Name). For a description of indirect references, see page 119.
- A variable must be globally scoped and published (internal or external) to be used in a C block.
- Symbolic variables cannot be used in the COMM_REQ status word.
- Use of symbolic variables is not supported on web pages.
- Symbolic Boolean variables are not allowed on non-BOOL parameters.
- Symbolic non-discrete variables cannot be used on Transitional contacts and coils. (Symbolic discrete variables are supported.)
- Overrides and Forces cannot be used on symbolic non-discrete variables. (Symbolic discrete variables are supported.)
- If Enhanced Security is enabled for a target, external devices such as HMIs and other controllers cannot access a symbolic variable unless the variable's Publish property is set to External Read/Write or External Read-Only.
- Arrays of the following data types are not supported:

  Arrays of user defined function block (UDFB) instance variables.

  Arrays of TON, TOF, or TP instance variables.

  Arrays of reference ID variables (RIVs) that contain one or more linked RIV elements.

  Note: A RIV array is supported when none of its elements is linked.

7.1.3 I/O Variables

An I/O variable is a symbolic variable that is mapped to a terminal in the hardware configuration. A terminal can be an actual physical discrete or analog I/O point or mapped to status data. The use of I/O variables allows you to configure hardware modules without having to specify the reference addresses to use when scanning their inputs and outputs. Instead, you can directly associate variable names with a module's inputs and outputs.

As with symbolic variables, memory required to support I/O variables counts against user space. You can configure the space available for I/O variables in the Memory tab of the PACSystems CPU.

For a given module, you must use either I/O variables or manually located mapped variables: you cannot use both in combination. It is not necessary to map all points on a module. Points that are disconnected or unused can be skipped. When points are skipped, space is reserved

in user memory for that point (that is, a 32-point discrete module will always use 32 bits of memory).

The hardware configuration (HWC) and logic become coupled in a PACSystems target on your computer as soon as you do one of the following: Enable I/O variables for a (even if you don't create any I/O variables) or upload a coupled HWC and logic from a PACSystems controller. The HWC and logic become coupled in a PACSystems controller when coupled HWC and logic are downloaded to it.

Effects of coupled HWC and logic:

- Whether the HWC and logic are coupled in the PACSystems target on your computer or in the PACSystems controller, you cannot download or upload the HWC and logic independently.
- When the HWC and logic are coupled in the PACSystems controller, you cannot clear the HWC and the logic independently.
- As for any download, you cannot run mode store (RMS) the HWC and logic independently.
- The HWC must be completely equal for you to make word-for-word changes, launch the Online Test mode of Test Edit, or accept the edits of Test Edit.

I/O variables can be used any place that other symbolic variables are supported, such as in logic as parameters to built-in function blocks, user defined function blocks, parameterized function blocks, C blocks, bit-in-word references, and transitional contacts and coils.

7.1.3.1 Restrictions on the Use of I/O Variables

- Since I/O variables are a form of symbolic variable, the same restrictions that apply to other symbolic variables of the same data type and array bounds apply to I/O variables.
- Only a global variable can become an I/O variable. A local variable cannot become an I/O variable.
- You can map only a discrete variable to a discrete terminal.
- You can map only a non-discrete variable to an analog terminal.
- Arrays and UDT variables must fit on the number of terminals in the reference address node counting from and including the terminal where you enter the array head or UDT variable. For example, if you have 32 analog terminals and you have a WORD array of 12 elements, you can map it to terminal 21 or any terminal before it (1 through 20).
- You can map a discrete array only to a terminal 8n+1, where n = 0, 1, 2, and so on. The "+1" is there because the terminals are numbered beginning with 1. If you map it to a terminal other than 8n+1, an error occurs upon validation.
- An I/O variable cannot be mapped to more than one location in hardware configuration.
- For the DO_IO function block, if an I/O variable is assigned to the ST parameter, then the same I/O variable must also be assigned to the END parameter, and the entire module is scanned.

7.1.3.2 I/O Variable Format

You create I/O variables by assigning them on the Terminals tab in PME Hardware Configuration. The variable structure is created automatically, and uses the format %vdr.s.[z.]g.t, where:

v = I (input) or Q (output)

d = data type: X (discrete) or W (analog).

[z] = subslot number. Always 0 for RXi Controllers.

g = segment number. Always 1 for the RXi Ethernet interface; always 2 for the RXi PNC.

t = terminal number. One-based, that is, the numbering begins at 1.

Supported I/O Variable Types

Data Type Mnemonic	*Supported Data Types*	*Number of Consecutive Terminals Required*
X	BOOL variable	1
	BOOL array	Number of elements in array.
	BYTE variable	8
	BYTE array	8n, where n is the number of array elements.
W	DINT variable	2
	DINT array	Number of elements in array times 2
	DWORD variable	2
	DWORD array	Number of elements in array times 2
	INT variable	1
	INT array	Number of elements in array
	LREAL variable	4
	LREAL array	Number of elements in array times 4
	REAL variable	2
	REAL array	Number of elements in array times 2
	UINT variable	1
	UINT array	Number of elements in array
	WORD variable	1
	WORD array	Number of elements in array

7.1.4 Arrays

An array is a complex data type composed of a series of variable elements with identical data types. Any variable can become an array, except for another array, a variable element, or a UDFB. In Machine Edition, you can create single-dimensional arrays and two-dimensional arrays.

In the controller CPU, each element of an array is treated as a separate variable with a separate, read-only reference address. The "root" node of the array variable also has a reference address that is editable. When you set or change the reference address of the "root" node of an array variable, the reference addresses of its elements are filled in with a range of addresses starting at that reference address and incremented for each element so as to create contiguous non-overlapping memory.

7.1.5 Variable Indexes and Arrays

The RXi Controller supports variable indexes for arrays. With a variable index, when logic is executed, the value of the variable is evaluated and the corresponding array element is accessed.

Note: The numbering of array elements is zero-based.

For example, to access an element of the array named ABC, you could write ABC[DEF] in logic. When logic is executed, if the value of DEF is 5, then ABC[DEF] is equivalent to ABC[5], and the sixth element of array ABC is accessed.

If the value of the variable index exceeds the array boundary, a non-fatal fault is logged to the Controller Fault table. In LD, the instruction for which this occurred does not pass power to the right.

Requirements and Support

An index variable must be of the INT, UINT, or DINT data type.

The valid range of values for an index variable is 0 through Y, where Y = [the number of array elements in the array] - 1. Ensuring that a variable index does not exceed the upper boundary of an array.

An index variable can be one of the following:

- Symbolic variable
- I/O variable
- Variable mapped to % memory areas such as %R
- Structure element
- Array element with a constant index
- Array element with a variable index
- Alias variable
- In the logic of a UDFB or parameterized block: formal parameter

The following support a variable index:

- Array elements of any data type except STRING
- Parameter array elements of any data type
- Alias variables

Dimensional support:

- One-dimensional (1D) formal parameter arrays in the logic of a UDFB or parameterized block
- 2D support for the top level of an array of structures and 1D support for a structure element that is an array. For example:

  PQR[a, b].STRU[y].Zed,

  where Zed is an element of the array of structures STRU, which itself is an element of the 2D array of structures PQR.
- 1D and 2D arrays for other variables

Other features:

- An array with a variable index supports a bit reference, for example

  MyArray[nIndex].X[4],

  where .X[4] is the fifth bit of the value stored in MyArray[nIndex]. The bit reference itself, [4] in the example, must be a constant.
- In LD, the following word-for-word changes are supported for array elements with variable indexes:

  Replacing an index variable with another index variable

  Replacing an index variable with a constant

  Replacing a constant with an index variable

  In LD, Diagnostic Logic Blocks support the use of array elements with variable indexes.

The following do not support array elements with variable indexes:

- Indirect references
- Reference ID variables (RIVs) and I/O variables when accessed in the Hardware Configuration

  Note: In logic, RIVs and I/O variables support variable indexes.
- STRING variables

A variable index cannot be one of the following:

- A math expression. For example, ABC[GH+1] is not supported.
- An indirect reference. For example, W[@XYZ] is not supported.
- A bit reference. For example, ABC[DEF.X[3]] is not supported.

  Note: You can use a bit reference on an array element designated by a variable index. For example, ABC[DEF]**.X[3]** is supported.
- An array head. For example, if MNP and QRS are arrays, MNP[QRS] is not supported, but MNP[QRS[3]] and MNP[QRS[TUV]] are, where TUV is an index variable.
- A negative index. This generates a run-time non-fatal CPU fault.
- A value greater than Y, where Y = [number of array elements] - 1. This generates a run-time non-fatal CPU fault.

7.1.6 Ensuring that a Variable Index does not Exceed the Upper Boundary of an Array

One-Dimensional Array

1. Once per scan, execute ARRAY_SIZE_DIM1 to count the number of elements in the array.

   Note: The array size of a variable can be changed in a run mode store but it will not be changed while logic is executing.

   ARRAY_SIZE_DIM1 places the count value in the variable associated with its output Q.
2. Before executing an instruction that uses a variable index, compare the value of the index variable with the number of elements in the array.

Tip: In LD, use a RANGE instruction.

Notes Checking before executing each instruction that uses an indexed variable is recommended in case logic has modified the index value beyond the array size or in case the array size has been reduced before the scan to less than the value of an index variable that has not been reduced accordingly since.

Valid range of an index variable: 0 through (n–1), where n is the number of array elements. Array indexes are zero-based.

Two-Dimensional Array

- Execute both ARRAY_SIZE_DIM1 and ARRAY_SIZE_DIM2 to count the number of elements in respectively the first and second dimensions of the array.

7.2 Reference Memory

The CPU stores program data in bit memory and word memory. Both types of memory are divided into different types with specific characteristics. By convention, each type is normally used for a specific type of data, as explained below. However, there is great flexibility in actual memory assignment.

Memory locations are indexed using alphanumeric identifiers called references. The reference's letter prefix identifies the memory area. The numerical value is the offset within that memory area, for example %AQ0056.

7.2.1 Word (Register) References

Type	Description
%AI	The prefix %AI represents an analog input register. An analog input register holds the value of one analog input or other non-discrete value.
%AQ	The prefix %AQ represents an analog output register. An analog output register holds the value of one analog output or other non-discrete value.
%R	Use the prefix %R to assign system register references that will store program data such as the results of calculations.
%W	Retentive Bulk Memory Area, which is referenced as %W (WORD memory).
%P	Use the prefix %P to assign program register references that will store program data with the _MAIN block. This data can be accessed from all program blocks. The size of the %P data block is based on the highest %P reference in all blocks. %P addresses are available only to the LD program they are used in, including C blocks called from LD blocks; they are not system-wide.

Note: All register references are retained across a power cycle to the CPU.

7.2.1.1 Indirect References

An indirect reference allows you to treat the contents of a variable assigned to an LD instruction operand as a pointer to other data, rather than as actual data. Indirect references are used only with word memory areas (%R, %W, %AI, %AQ, %P, and %L). An indirect reference in %W requires two %W locations as a DWORD indirect index value. For example, @%W0001 would use the %W2:W1 as a DWORD index into the %W memory range. The DWORD index is required because the %W size is greater than 65K.

Indirect references cannot be used with symbolic variables.

To assign an indirect reference, type the @ character followed by a valid reference address or variable name. For example, if %R00101 contains the value 1000, @R00101 instructs the CPU to use the data location of %R01000.

Indirect references can be useful when you want to perform the same operation to many word registers. Use of indirect references can also be used to avoid repetitious logic within the application program. They can be used in loop situations where each register is incremented by a constant or by a value specified until a maximum is reached.

7.2.1.2 Bit in Word References

Bit in word referencing allows you to specify individual bits in a word reference type as inputs and outputs of Boolean expressions, functions, and calls that accept bit parameters (such as parameterized blocks). This feature is restricted to word references in retentive memory. The bit number in the bit within word construct must be a constant.

You can use the programmer or an HMI to set an individual bit on or off within a word, or monitor a bit within a word. Also, C blocks can read, modify, and write a bit within a word.

Bit in Word references can be used in the following situations:

- In retentive 16-bit memory (AI, AQ, R, W, P, and L) and symbolics.
- On all contacts and coils ***except*** legacy transition contacts (POSCON/NEGCON) and transition coils (POSCOIL/NEGCOIL).
- On all functions and call parameters that accept single or unaligned bit parameters.

Functions that accept unaligned discrete references	***Parameters***
ARRAY MOVE (BIT)	SR and DS
ARRAY RANGE (BIT)	Q
MOVE (BIT)	IN and Q
SHFR (BIT)	IN, ST and Q

The use of Bit in Word references has the following restrictions:

- Bit in Word references cannot be used on legacy transition contacts (POSCON/NEGCON) and transition coils (POSCON/NEGCON).
- The bit number (index) must be a constant; it cannot be a variable.
- Bit addressing is not supported for a constant.
- Indirect references cannot be used to address bits in 16-bit memory.
- You cannot force a bit within 16-bit memory.

Examples: :

%R2.X [0] addresses the first (least significant) bit of %R2

%R2.X [1] addresses the second bit of %R2. In the examples

In the examples [0] and [1] are the bit indexes. Valid bit indexes for the different variable types are:

BYTE variable	[0] through [7]
WORD, INT, or UINT variable	[0] through [15]
DWORD or DINT variable	[0] through [31]

7.2.2 Bit (Discrete) References

Type	Description
%I	Represents input references. %I references are located in the input status table, which stores the state of all inputs received from input modules during the last input scan. A reference address is assigned to discrete input modules using your programming software. Until a reference address is assigned, no data will be received from the module. %I memory is always retentive.
%Q	Represents physical output references. The coil check function checks for multiple uses of %Q references with relay coils or outputs on functions. You can select the level of coil checking desired (Single, Warn Multiple, or Multiple). %Q references are located in the output status table, which stores the state of the output references as last set by the application program. This output status table's values are sent to output modules at the end of the program scan. A reference address is assigned to discrete output modules using your programming software. Until a reference address is assigned, no data is sent to the module. A particular %Q reference may be either retentive or non-retentive.
%M	Represents internal references. The coil check function of your programming software checks for multiple uses of %M references with relay coils or outputs on functions. A particular %M reference may be either retentive or non-retentive.
%T	Represents temporary references. These references are never checked for multiple coil use and can, therefore, be used many times in the same program even when coil use checking is enabled—this is not a recommended practice because it makes subsequent trouble-shooting more difficult. %T may be used to prevent coil use conflicts while using the cut/paste and file write/include functions. Because this memory is intended for temporary use, it is cleared on Stop-to-Run transitions and cannot be used with retentive coils.
%S %SA %SB %SC	Represent system status references. These references are used to access special CPU data such as timers, scan information, and fault information. For example, the %SC0012 bit can be used to check the status of the Controller Fault table. Once the bit is set on by an error, it will not be reset until after the sweep. %S, %SA, %SB, and %SC can be used on any contacts. ■ %SA, %SB, and %SC can be used on retentive coils -(M)-. **Note:** Although the programming software forces the logic to use retentive coils with %SA, %SB, and %SC references, most of these references are not preserved across power cycles regardless of the state of the Energy Pack if one exists. %S can be used as word or bit-string input arguments to functions or function blocks. %SA, %SB, and %SC can be used as word or bit-string input or output arguments to functions and function blocks. For a description of the behavior of each bit, see "System Status References" on page 125.
%G	Represents global data references. These references are used to access data shared among several control systems.

Note: For details on retentiveness, refer to "Retentiveness of Logic and Data" on page 123.

7.3 User Reference Size and Default

Maximum user references and default reference sizes are listed in the table below.

Item	Range	Default
Reference Points		
%I reference	32768 bits	32768 bits
%Q reference	32768 bits	32768 bits
%M reference	32768 bits	32768 bits
%S total (S, SA, SB, SC)	512 bits (128 each)	512 bits (128 each)
%T reference	1024 bits	1024 bits
%G	7680 points	7680 points
Total Reference Points	107520	107520
Reference Words		
%AI reference	0—32640 words	64 words
%AQ reference	0—32640 words	64 words
%R, 1K word increments	0—32640 words	1024 words
%W	0—maximum available user RAM	0 words
Total Reference Words	0—maximum available user RAM	1152 words
%L (per block)	8192 words	8192 words
%P (per program)	8192 words	8192 words
Managed Memory		
Symbolic Discrete	0—83,886,080 (bits)	32768
Symbolic Non-Discrete	0—5,242,880 (words)	65536
I/O Discrete	0 through 83,886,080	0
I/O Non-Discrete	0 through 5,242,880.	0
Total Symbolic	0—42,088,704 bytes (This is the total memory available for the combined total of symbolic memory. This also includes other user memory use, program etc.)	143360

7.3.1 %G User References and CPU Memory Locations

The CPU contains one data space for all of the global data references (%G). The internal CPU memory for this data is 7680 bits long.

7.4 Transitions and Overrides

The %I, %Q, %M, and %G user references, and symbolic variables of type BOOL, have associated transition and override bits. %T, %S, %SA, %SB, and %SC references have transition bits but not override bits. The CPU uses transition bits for counters, transitional contacts, and transitional coils. Note that counters do not use the same kind of transition bits as contacts and coils. Transition bits for counters are stored within the locating reference.

The transition bit for a reference tells whether the most recent value (ON, OFF) written to the reference is the same as the previous value of the reference. Therefore when a reference is written and its new value is the same as its previous value, its transition bit is turned OFF. When its new value is different from its previous value, its transition bit is turned ON. The transition bit for a reference is affected every time the reference is written to. The source of the write is immaterial; it can result from a coil execution, an executed function's output, the updating of reference memory after an input scan, etc.

When override bits are set, the associated references cannot be changed from the program or the input device; they can only be changed on command from the programmer. Overrides do not protect transition bits. If an attempted write occurs to an overridden memory location, the corresponding transition bit is cleared.

7.5 Retentiveness of Logic and Data

Data is defined as retentive if it is saved by the CPU when the CPU transitions from STOP mode to RUN mode.

The following items are retentive:

- program logic
- fault tables and diagnostics
- checksums for program logic
- overrides and output forces
- word data (%R, %W, %L, %P, %AI, %AQ)
- bit data (%I, %G, and reserved bits)
- %Q and %M variables that are configured as retentive (%T data is non-retentive and therefore not saved on STOP to RUN transitions.)
- symbolic variables that have a data type other than BOOL
- symbolic variables of BOOL type that are configured as retentive
- Retentive data is also preserved during power-cycles of the CPU with Energy Pack backup. Exceptions to this rule include most of the %S, %SA, %SB, and %SC references. These references are initialized to zero at power-up regardless of the state of the Energy Pack if one exists. (See page 125 for a description of the behavior of each system status reference.)

When %Q or %M variables are configured as retentive, the contents are retained through power loss and Run-to-Stop-to-Run transitions.

7.6 Data Scope

Each of the user references has “scope”; that is, it may be available throughout the system, available to all programs, restricted to a single program, or restricted to local use within a block.

User Reference Type	Range	Scope
%I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %W, %AI, %AQ, convenience references	Global	From any program, block, or host computer. Variables defined in these registers have system (global) scope by default. However, variables with local scope can also be assigned in these registers.
Symbolic variable	Global	From any program, block, or host computer. Symbolic variables have system (global) scope by default. However, symbolic variables with local scope can be created using the naming conventions for local variables.
I/O variable	Global	From any program, block, or host computer.
%P	Program	From any block, but not from other programs (also available to a host computer).
%L	Local	From within a block only (also available to a host computer).

In an LD block:

- %P should be used for program references that are shared with other blocks.
- %L are local references that can be used to restrict the use of register data to that block. These local references are not available to other parts of the program.
- %I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %W, %AI, and %AQ references are available throughout the system.

7.7 System Status References

System status references in the CPU are assigned to %S, %SA, %SB, and %SC memory. The four timed contacts (time tick references) include #T_10MS, #T_100MS, #T_SEC, and #T_MIN. Examples of other system status references include #FST_SCN, #ALW_ON, and #ALW_OFF

Note: %S bits are read-only bits; do not write to these bits. You may, however, write to %SA, %SB, and %SC bits.

Listed below are available system status references that may be used in an application program. When entering logic, either the reference or the nickname can be used. Refer to "Chapter 8, Diagnostics" for detailed fault descriptions and information on correcting faults.

7.7.1 %S References

Reference	Name	Definition
%S0001	#FST_SCN	Current sweep is the first sweep in which the LD executed. Set the first time the user program is executed after a Stop/Run transition and cleared upon completion of its execution.
%S0002	#LST_SCN	Set when the CPU transitions to run mode and cleared when the CPU is performing its final sweep. The CPU clears this bit and then performs one more complete sweep before transitioning to Stop or Stop Faulted mode. If the number of last scans is configured to be 0, %S0002 will be cleared after the CPU is stopped and user logic will not see this bit cleared.
%S0003	#T_10MS	0.01 second timed contact.
%S0004	#T_100MS	0.1 second timed contact.
%S0005	#T_SEC	1.0 second timed contact.
%S0006	#T_MIN	1.0 minute timed contact.
%S0007	#ALW_ON	Always ON.
%S0008	#ALW_OFF	Always OFF.
%S0009	#SY_FULL	Set when the Controller Fault table fills up (size configurable with a default of 16 entries). Cleared when an entry is removed from the Controller Fault table and when the fault table is cleared.
%S0010	#IO_FULL	Set when the I/O fault table fills up (size configurable with a default of 32 entries). Cleared when an entry is removed from the I/O fault table and when the I/O fault table is cleared.
%S0011	#OVR_PRE	Set when an override exists in %I, %Q, %M, or %G, or symbolic BOOL memory.
%S0012	#FRC_PRE	Reserved.
%S0013	#PRG_CHK	Set when background program check is active.
%S0014	#PLC_BAT	Energy Pack is connected and functioning = 0 Energy Pack is not connected or has failed = 1

Note: The #FST_EXE name is not associated with a %S address; it must be referenced by the name "#FST_EXE" only. This bit is set when transitioning from Stop to Run and indicates that the current sweep is the first time this block has been called.

7.7.2 %SA, %SB, and %SC References

Note: %SA, %SB, and %SC contacts are not set or reset until the input scan phase of the sweep following the occurrence of the fault or a clearing of the fault table(s). %SA, %SB, and %SC contacts can also be set or reset by user logic and CPU monitoring devices.

Reference	Name	Definition
%SA0001	#PB_SUM	Set when a checksum calculated on the application program does not match the reference checksum. If the fault was due to a temporary failure, the condition can be cleared by storing the program again to the CPU. If the fault was due to a memory failure, the Controller must be replaced. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SA0002	#OV_SWP	Set when the CPU detects that the previous sweep took longer than the time specified by the user. To clear this bit, clear the Controller Fault table or power cycle the Controller. Only occurs if the Controller is in Constant Sweep mode.
%SA0003	#APL_FLT	Set when an application fault occurs. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SA0009	#CFG_MM	Set when a configuration mismatch fault is logged in the fault tables. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SA0008	#OVR_TMP	Set when the operating temperature of the Controller exceeds the normal operating temperature. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SA0010	#HRD_CPU	Set when the diagnostics detects a problem with the CPU hardware. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SA0013	#LOS_IOC	Set when a Bus Controller stops communicating with the CPU. To clear this bit, clear the I/O fault table or power cycle the Controller.
%SA0014	#LOS_IOM	Set when an I/O module stops communicating with the CPU. To clear this bit, clear the I/O fault table or power cycle the Controller.
%SA0015	#LOS_SIO	Reserved.
%SA0022	#IOC_FLT	Set when a Bus Controller reports a bus fault, a global memory fault, or an IOC hardware fault. To clear this bit, clear the I/O fault table or power cycle the Controller.
%SA0023	#IOM_FLT	Set when an I/O module reports a circuit or module fault. To clear this bit, clear the I/O fault table or power cycle the Controller.
%SA0027	#HRD_SIO	Reserved.
%SA0029	#SFT_IOC	Set when there is a software failure in the embedded PNC. To clear this bit, clear the I/O Fault table or power cycle the Controller.
%SA0081 – %SA0112		Set when a user-defined fault is logged in the Controller Fault table. To clear these bits, clear the Controller Fault table or power cycle the Controller. User-defined faults are created using SVC_REQ 21.
%SB0001	#WIND_ER	Set when there is not enough time to start the Programmer Window in Constant Sweep mode. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SB0009	#NO_PROG	Set when the CPU powers up with memory preserved, but no user program is present. Cleared when the CPU powers up with a program present or by clearing the Controller Fault table.
%SB0010	#BAD_RAM	Set when the CPU detects corrupted user memory at power-up. Cleared when the CPU detects that user memory is valid at power-up or by clearing the Controller Fault table.
%SB0011	#BAD_PWD	Set when a password access violation occurs. Cleared when the Controller Fault table is cleared or when the Controller is power cycled.

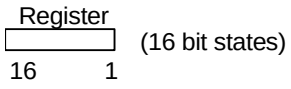
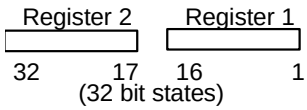
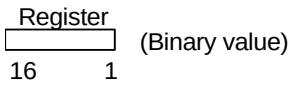
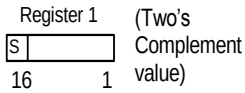
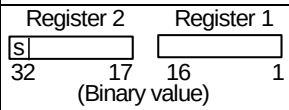
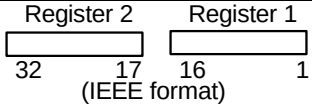
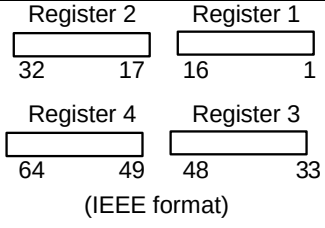
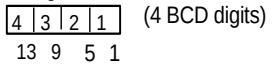
Reference	Name	Definition
%SB0012	#NUL_CFG	Set when an attempt is made to put the CPU in Run mode when there is no configuration data present. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SB0013	#SFT_CPU	Set when the CPU detects an error in the CPU operating system software. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SB0014	#STOR_ER	Set when an error occurs during a programmer store operation. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SB0016	#MAX_IOC	Set when more than 32 IOCs are configured for the system. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SB0017	#SBUS_FL	Set when the CPU fails to gain access to the bus. To clear this bit, clear the Controller Fault table or power cycle the Controller.
%SC0009	#ANY_FLT	Set when any fault occurs that causes an entry to be placed in the CPU or I/O fault table. Cleared when both fault tables are cleared or when the Controller is power cycled.
%SC0010	#SY_FLT	Set when any fault occurs that causes an entry to be placed in the Controller Fault table. Cleared when the Controller Fault table is cleared or when the Controller is power cycled.
%SC0011	#IO_FLT	Set when any fault occurs that causes an entry to be placed in the I/O fault table. Cleared when the I/O fault table is cleared or when the Controller is power cycled.
%SC0012	#SY_PRES	Set as long as there is at least one entry in the Controller Fault table. Cleared when the Controller Fault table is cleared.
%SC0013	#IO_PRES	Set as long as there is at least one entry in the I/O fault table. Cleared when the I/O fault table is cleared.
%SC0014	#HRD_FLT	Set when a hardware fault occurs. Cleared when both fault tables are cleared or when the Controller is power cycled.
%SC0015	#SFT_FLT	Set when a software fault occurs. Cleared when both fault tables are cleared or when the Controller is power cycled.

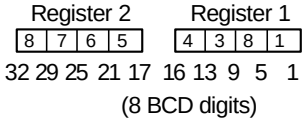
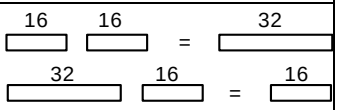
Fault references consist of System Fault References, Configurable Fault References and Non-Configurable Fault References. These are discussed in “8.3, System Handling of Faults.”

7.8 How Program Functions Handle Numerical Data

Regardless of where data is stored in memory – in one of the bit memories or one of the word memories – the application program can handle it as different data types.

7.8.1 Data Types

Type	Name	Description	Data Format
BOOL	Boolean	The smallest unit of memory. Has two states, 1 or 0. A BOOL array may have length N.	
BYTE	Byte	Has an 8-bit value. Has 256 values (0–255). A BYTE array may have length N.	
WORD	Word	Uses 16 consecutive bits of data memory. The valid range of word values is 0000 hex to FFFF hex.	
DWORD	Double Word	Has the same characteristics as a single word data type, except that it uses 32 consecutive bits in data memory instead of only 16 bits.	
UINT	Unsigned Integer	Uses 16-bit memory data locations. They have a valid range of 0 to +65535 (FFFF hex).	
INT	Signed Integer	Uses 16-bit memory data locations, and are represented in 2's complement notation. The valid range of an INT data type is –32768 to +32767.	 s=sign bit (0=positive, 1=negative)
DINT	Double Precision Integer	Stored in 32-bit data memory locations (two consecutive 16-bit memory locations). Always signed values (bit 32 is the sign bit). The valid range of a DINT data type is -2147483648 to +2147483647	 s=sign bit (0=positive, 1=negative)
REAL	Floating Point	Uses 32 consecutive bits (two consecutive 16-bit memory locations). The range of numbers that can be stored in this format is from ±1.401298E-45 to ±3.402823E+38. For the IEEE format, refer to "Floating Point Numbers" on page 130.	
LREAL	Double Precision Floating Point	Uses 64 consecutive bits (four consecutive 16-bit memory locations). The range of numbers that can be stored in this format is from ±2.2250738585072020E-308 to ±1.7976931348623157E+308. For the IEEE format, refer to "Floating Point Numbers" on page 130.	
BCD-4	Four-Digit BCD	Uses 16-bit data memory locations. Each binary coded decimal (BCD) digit uses four bits and can represent numbers between 0 and 9. This BCD coding of the 16 bits has a legal value range of 0 to 9999.	

Type	Name	Description	Data Format
BCD-8	Eight-Digit BCD	Uses two consecutive 16-bit data memory locations (32 consecutive bits). Each BCD digit uses 4 bits per digit to represent numbers from 0 to 9. The complete valid range of the 8-digit BCD data type is 0 to 99999999.	
MIXED	Mixed	Available only with the MUL and DIV functions. The MUL function takes two integer inputs and produces a double integer result. The DIV function takes a double integer dividend and an integer divisor to product an integer result.	
ASCII	ASCII	Eight-bit encoded characters. A single word reference is required to make two (packed) ASCII characters. The first character of the pair corresponds to the low byte of the reference word. The remaining 7 bits in each section are converted.	

Note: Using functions that are not explicitly bit-typed will affect transitions for all bits in the written byte/word/dword. For information about using floating point numbers, refer to “Floating Point Numbers” on page 130.

7.8.2 Floating Point Numbers

Floating point numbers are stored in one of two IEEE 754 standard formats that uses adjacent 16-bit words: 32-bit single precision or 64-bit double precision.

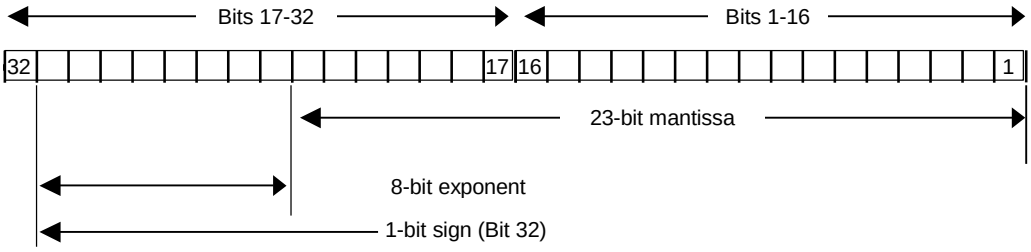
The REAL data type represents single precision floating point numbers. The LREAL data type represents double precision floating point numbers. REAL and LREAL variables are typically used to store data from analog I/O devices, calculated values, and constants.

Types of Floating Point Variables

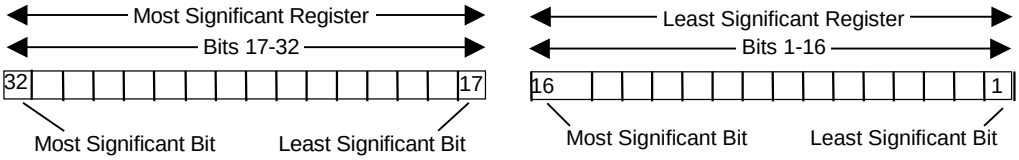
Data Type	*Precision and Range*
REAL	Limited to 6 or 7 significant digits, with a range of approximately $\pm 1.401298 \times 10^{-45}$ through $\pm 3.402823 \times 10^{38}$.
LREAL	Limited to 17 significant digits, with a range of approximately $\pm 2.2250738585072020 \times 10^{-308}$ to $\pm 1.7976931348623157 \times 10^{308}$.

Note: The programming software allows 32-bit and 64-bit arguments (DWORD, DINT, REAL, and LREAL) to be placed in discrete memories such as %I, %M, and %R in the PACSystems target.

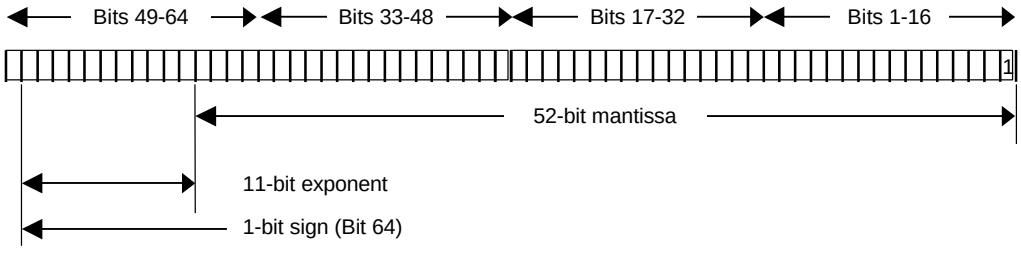
7.8.2.1 Internal Format of REAL Numbers



Register use by a single floating point number is diagrammed below. For example, if the floating point number occupies registers R5 and R6, R5 is the least significant register and R6 is the most significant register.



7.8.2.2 Internal Format of LREAL Numbers



7.8.2.3 Errors in Floating Point Numbers and Operations

Overflow occurs when a REAL or LREAL function generates a number outside the allowed range. When this occurs, the Enable Out output of the function is set Off, and the result is set to positive infinity (for a number greater than the upper limit) or negative infinity (for a number less than the lower limit). You can determine where this occurs by testing the sense of the Enable Out output.

Binary representations of Infinity and NaN values have exponents that contain all 1s.

IEEE 754 Infinity Representations

	REAL	LREAL
POS_INF (positive infinity)	= 7F800000h	= 7FF0000000000000h
NEG_INF (negative infinity)	= FF800000h	= 7FF0000000000001h

If the infinities produced by overflow are used as operands to other REAL or LREAL functions, they may cause an undefined result. This undefined result is referred to as an NaN (Not a Number). For example, the result of adding positive infinity to negative infinity is undefined. When the ADD_REAL function is invoked with positive infinity and negative infinity as its operands, it produces an NaN. If any operand of a function is a NaN, the result will be some NaN.

Note: For NaN, the Enable Out output is Off (not energized).

IEEE 754 Representations of NaN values:

REAL	LREAL
7F800001 through 7FFFFFFF	7FF8000000000001 through 7FFFFFFFFFFFFFFF
FF800001 through FFFFFFFF	FFF0000000000001 through FFFFFFFFFFFFFFFF

Note: For releases 5.0 and greater, the CPU may return slightly different values for NaN compared to previous releases. In some cases, the result is a special type of NaN displayed as #IND in Machine Edition. In these cases, for example, EXP(-infinity), power flow out of the function is identical to that in previous releases.

7.9 User Defined Types:

A UDT is a structured data type consisting of elements of other selected data types. Each top-level UDT element can be one of the following:

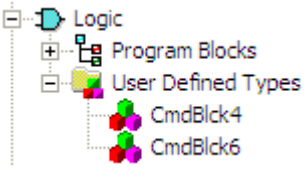
Top-level UDT Element	Example
Simple data type, except STRING	INT
Another UDT, except any in which the current UDT is nested at any level. Note: A UDT cannot be nested within itself.	A UDT named UDT_ABC has a top-level element whose data type is another UDT, named UDT_2.
Array of a simple data type	LREAL array of length 8.
Array of UDTs Note: A UDT cannot be nested within itself.	A UDT named UDT_ABC has a top-level element that is an array whose data type is another UDT, named UDT_row.

7.9.1 Working with UDTs

1. In PROFICY Machine Edition, add a UDT as a node under a target in the Project tab of the Navigator.

   A UDT is saved with the target it's used in.

2. Edit the UDT properties and define the elements in the UDT's structure.



3. Create a variable whose data type is the UDT.

   By default, the variable resides in symbolic memory. You can convert the symbolic variable to an I/O variable by assigning it to an I/O terminal.

4. Use the variable in logic.

7.9.2 UDT Properties

Name: The UDT's name. Maximum length: 32 characters.

Description: The user-defined description of the UDT.

Memory Type: The type of symbolic or I/O variable memory in which a variable of this UDT resides.

Non-Discrete: (Default) Word-oriented memory organized in groups of 16 contiguous bits.

Discrete: Bit-oriented memory.

Notes: You cannot nest a UDT of one memory type in a UDT of a different memory type.

Changing the memory type propagates to existing variables of this UDT only after target validation.

Is Fixed Size: If set to True, you can increase the Size (Bytes) value to a maximum of 65,535 bytes to create a buffer at the end of the UDT. The buffer is included in the memory allocated to every downloaded variable of that UDT data type. Use of a buffer may allow run mode store of a UDT when the size of the UDT definition has changed. For details, see page 133.

If set to False (default), the Size (Bytes) value is read-only and does not include a buffer at the end of the UDT.

Size (bytes): (Read-only when Is Fixed Size is set to False.) The total number of bytes required to store a structure variable of the user-defined data type (UDT).

Bytes Remaining: (Read-only; displayed if Is Fixed Size is set to True.) The UDT's buffer size; the number of bytes available before the actual size of the UDT reaches the value of the Size (bytes) property.

7.9.3 UDT Limits

- Maximum number of UDTs per target: 2048
- Maximum UDT size: 65,535 bytes

  Note: Bit spares created to line up the end of a section of BOOL variables or arrays with the end of a byte will count toward the maximum size.
- Maximum number of top-level UDT elements: 1024
- Maximum array size of a top-level UDT element: 1024 array elements
- UDTs ***do not*** support the following:
  - Two-dimensional arrays
  - Function block data types
  - Enumerated data types
- You cannot nest a UDT of one memory type in a UDT of a different memory type.
- You cannot alias a variable to a UDT variable or UDT variable element.
- A FAULT contact supports a BOOL element of a UDT I/O variable, but not a BOOL element of a UDT parameter in a UDFB or parameterized block.
- POSCON and NEGCON do not support BOOL elements of UDT parameters in parameterized blocks or UDFBs.

7.9.4 Run Mode Store of UDTs

An RMS can be performed on a target that contains a variable of a UDT, unless:

- An operation in the UDT editor modifies the offset or bit mask of an element that has the same name before and after the operation.
- The size of the UDT definition increases.
- Array length increases.
- The memory type of the UDT definition changes.
- There is a data type change in the UDT definition, except for the following interchangeable data types:
  - WORD, INT, UINT
  - DWORD, DINT
- The UDT definition is renamed.

7.9.5 UDT Operational Notes

- By default, a UDT variable resides in symbolic memory. You can convert the symbolic variable to an I/O variable.
- All UDT elements are public and, therefore, readable and writeable.
- Properties of elements of UDT variables:

  The Input Transfer List and Output Transfer List properties are read-only and set to False.

  The Retentive property is editable only for BOOLs and only if the UDT Memory Type is discrete. For UDTs whose Memory Type is non-discrete, a BOOL variable has its Retentive property set to True during validation.
- UDT variables are supported in LD, FBD, and ST blocks, as well as in Diagnostic Logic Blocks.

For additional operational notes, refer to the programmer Help.

7.10 Operands for Instructions

The operands for PACSystems instructions can be in the following forms:

- Constants
- Variables that are located in any of the PACSystems memory areas (%I, %Q, %M, %T, %G, %S, %SA, %SB, %SC, %R, %W, %L, %P, %AI, %AQ)
- Symbolic variables, including I/O variables
- Parameters of a Parameterized block or C block
- Power flow
- Data flow
- Computed references such as indirect references or bit-in-word references
- BOOL arrays

An operand's type and length must be compatible with that of the parameter it is being passed into. PACSystems instructions and functions have the following operand restrictions:

- Constants cannot be used as operands to output parameters because output values cannot be written to constants.
- Variables located in %S memory cannot be used as operands to output parameters because %S memory is read-only.
- Variables located in %S, %SA, %SB, and %SC memories cannot be used as operands to numerical parameters such as INTs, DINTs, REALs, LREALs, etc.
- Data flow is prohibited on some input parameters of some functions. This occurs when the function, during the course of its execution, actually writes a value to the input parameter. Data flow is prohibited in these cases because data flow is stored in a temporary memory and any updated value assigned to it would be inaccessible to the user application.
- The arguments to EN, OK, and many other BOOLEAN input and output parameters are restricted to be power flow.
- Restrictions on using Parameterized block or External block parameters as operands to instructions or functions are documented in "6.1.5.2, Parameterized Blocks."
- References in discrete memory (I, Q, M, and T) must be byte-aligned.

Note the following:

- Indirect references, which are available for all WORD-oriented memories (%R, %W, %P, %L, %AI, %AQ), can be used as arguments to instructions wherever located variables in the corresponding WORD-oriented memory are allowed. Note that indirect references are converted into their corresponding direct references immediately before they are passed into an instruction or function.
- Bit-in-word references are generally allowed on contact and coil instructions other than transition contacts and coils. They are also allowed as arguments to function parameters that accept single or unaligned bits.

BOOL arrays can be used as parameters to an instruction instead of variables of other data types. The array must be of sufficient length to replace the given data type. For example, instead of using a 16-bit INT variable, you could use a BOOL array of length 16 or more.

The following conditions must be met:

- The BOOL array must be byte-aligned, that is, the reference address of the first element of the BOOL array must be 8n + 1, where n = 0, 1, 2, 3, and so on. For example, %M00033 is byte-aligned, because 33 = (8 × 4) + 1.
- The parameter in question must support discrete memory reference addresses.
- The instruction in question must not have a Length parameter. (The Length parameter is displayed as ?? in the LD editor until a value is assigned.)
- The data type to be replaced with a BOOL array must be one of the following:

Data Type	*Minimum Length*
BYTE	8
INT, UINT, WORD	16
DINT, DWORD, REAL	32
REAL	64

- Excess bits are ignored. For example, if you use a BOOL array of length 12 instead of an 8-bit BYTE, the last four bits of the BOOL array are ignored.

7.11 Word-for-Word Changes

Many changes to the program that do not modify the size of the program are considered word-for-word changes. Examples include changing the type of contact or coil, or changing a reference address used for an existing function block.

Symbolic Variables

Creating, deleting, or modifying a symbolic variable definition is not a word-for-word change.

The following are word-for-word changes:

- Switching between two symbolic variables
- Switching between an symbolic variable and a mapped variable
- Switching between a constant and a symbolic variable

Chapter 8. Diagnostics

This chapter explains the PACSystems fault handling system, provides definitions of fault extra data, and suggests corrective actions for faults.

Faults occur in the control system when certain failures or conditions happen that affect the operation and performance of the system. Some conditions, such as the loss of an I/O module, may impair the ability of the PACSystems controller to control a machine or process. Other conditions, such as when a new module comes online and becomes available for use, may be displayed to inform or alert the user.

Any detected fault is recorded in the controller fault table or the I/O fault table, as applicable.

Information in this chapter is organized as follows:

- Fault Handling Overview 136
- Using the Fault Tables 138
- System Handling of Faults 142
- Controller Fault Descriptions and Corrective Actions 150
- I/O Fault Descriptions and Corrective Actions 150

8.1 Fault Handling Overview:

The PACSystems CPU detects three classes of faults:

Fault Class	Examples
Internal Failures (Hardware)	Non-responding modules Memory checksum errors
External I/O Failures (Hardware)	Loss of module Addition or module
Operational Failures	Communication failures Configuration failures Password access failures

8.1.1 System Response to Faults

Hardware failures require that either the system be shut down or the failure be tolerated. I/O failures may be tolerated by the control system, but they may be intolerable by the application or the process being controlled. Operational failures are normally tolerated.

Faults have three attributes:

Fault Table Affected	I/O fault table controller fault table
Fault Action	Fatal Diagnostic Informational
Configurability	Configurable Nonconfigurable

8.1.2 Fault Tables

The PACSystems CPU maintains two fault tables, the Controller Fault table for internal CPU faults and the I/O Fault table for faults generated by I/O devices (including I/O controllers). For more information, see “Using the Fault Tables” on page 138.

8.1.3 Fault Actions and Fault Action Configuration

Fatal faults cause the fault to be recorded in the appropriate table, diagnostic variables to be set, and the system to be stopped. Only fatal faults cause the system to stop.

Diagnostic faults are recorded in the appropriate table, and any diagnostic variables are set. Informational faults are only recorded in the appropriate table.

Fault Action	*Response by CPU*
Fatal	Log fault in fault table. Set fault references. Go to Stop/Fault mode.
Diagnostic	Log fault in fault table. Set fault references.
Informational	Log fault in fault table.

The hardware configuration can be used to specify the fault action of some fault groups. For these groups, the fault action can be configured as either fatal or diagnostic. When a fatal or diagnostic fault within a configurable group occurs, the CPU executes the configured fault action instead of the action specified within the fault.

Note: The fault action displayed in the expanded fault details indicates the fault action specified by the fault that was logged, but not necessarily the executed fault action. To determine what action was executed for a particular fault in a configurable fault group, you must refer to the hardware configuration settings.

Faults that are part of configurable fault groups:

Fault Action Displayed in Fault Table	Informational	Diagnostic	Fatal
Fault Action Executed	Informational	Diagnostic or Fatal. Determined by action selected in Hardware Configuration.	Diagnostic or Fatal. Determined by action selected in Hardware Configuration.

Faults that are part of nonconfigurable fault groups:

Fault Action Displayed in Fault Table	Informational	Diagnostic	Fatal
Fault Action Executed	Informational	Diagnostic	Fatal

8.2 Using the Fault Tables

To display the fault tables in Logic Developer software,

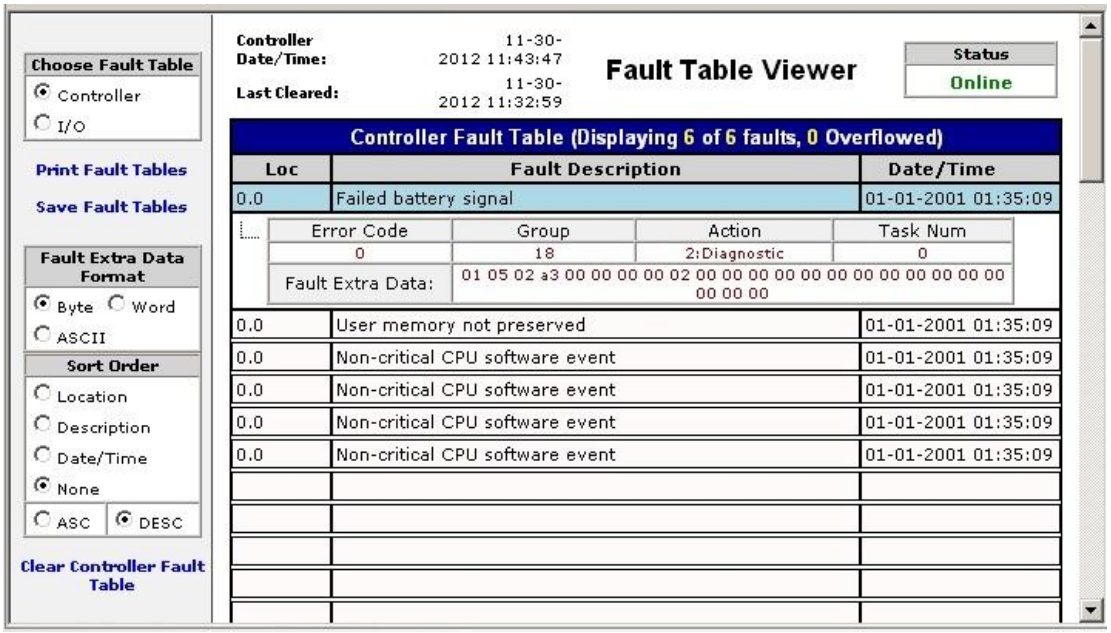
1. Go online with the PACSystems.
2. Select the Project tab in the Navigator, right click the Target node and choose Diagnostics. The Fault Table Viewer appears.

The controller fault table and the I/O fault table display the following information:

Controller Time/Date	The current date and time of the CPU.
Last Cleared	The date and time faults were last cleared from the fault table. This information is maintained by the PACSystems controller.
Status	Displays “Updating” while the programmer is reading the fault table. Status is “Online” when update is complete.
Total Faults	The total number of faults since the table was last cleared.
Entries Overflowed	The number of entries lost because the fault table has overflowed since it was cleared. Each fault table can contain up to 64 faults.

8.2.1 Controller Fault Table

The controller fault table displays CPU faults such as password violations, configuration mismatches, parity errors, and communications errors.



The controller fault table provides the following information for each fault:

Location Identifies the location of the fault by rack.slot (always 0.0 for the RXi Controller).

Description Corresponds to a fault *group*, which is identified in the fault Details.

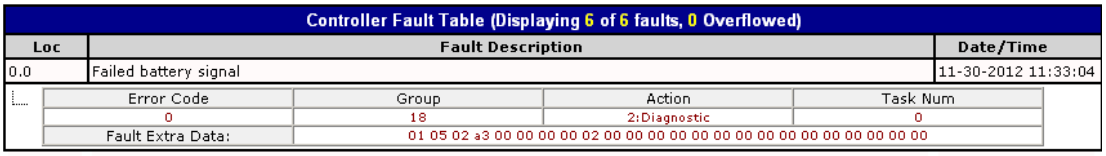
Date/Time The date and time the fault occurred based on the CPU clock.

Details To view detailed information, click the fault entry. See “Viewing Controller Fault Details” for more information.

8.2.1.1 Viewing Controller Fault Details

Note: The fault action displayed in the expanded fault details indicates the fault action specified by the fault that was logged, but not necessarily the executed fault action. To determine what action was executed for a particular fault in a configurable fault group, you must refer to the hardware configuration settings.

To see controller fault details, click the fault entry. The detailed information box for the fault appears. (To close this box, click the fault.)



Controller Fault Table (Displaying 6 of 6 faults, 0 Overflowed)		
Loc	Fault Description	Date/Time
0.0	Failed battery signal	11-30-2012 11:33:04

Error Code	Group	Action	Task Num
0	18	2:Diagnostic	0
Fault Extra Data:	01 05 02 a3 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

The detailed information for controller faults includes the following:

Error Code Further identifies the fault. Each fault group has its own set of error codes.

Group Group is the highest classification of a fault and identifies the general category of the fault. The fault description text displayed by your programming software is based on the fault group and the error codes.

Action Fatal, Diagnostic, or Informational. For definitions of these actions, refer to page 137.

Task Number Not used for most faults. When used, provides additional information for Technical Support representatives.

Fault Extra Data Provides additional information for diagnostics by Technical Support engineers. Explanations of this information are provided as appropriate for specific faults in “Controller Fault Descriptions and Corrective Actions” on page 150.

8.2.1.2 User Defined Faults

User-defined faults can be logged in the controller fault table. When a user-defined fault occurs, it is displayed in the appropriate fault table as “Application Msg (error_code):” and may be followed by a descriptive message up to 24 characters. The user can define all characters in the descriptive message. Although the message must end with the null character, e.g., zero (0), the null character does not count as one of the 24 characters. If the message contains more than 24 characters, only the first 24 characters are displayed.

Certain user-defined faults can be used to set a system status reference (%SA0081–%SA0112).

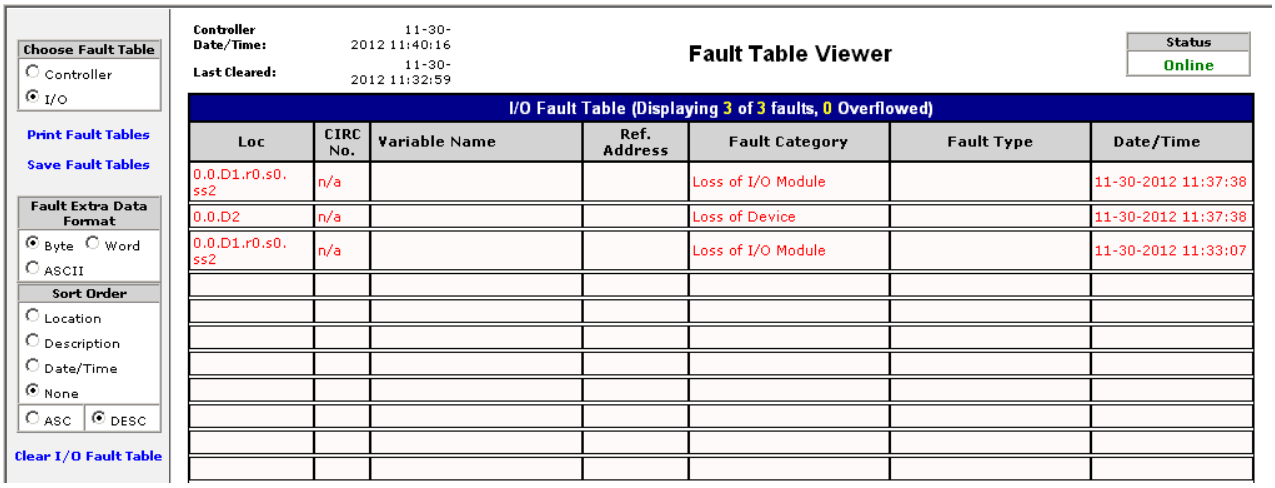
User-defined faults are created using Service Request 21.

Note: When a user-defined fault is displayed in the Controller Fault table, a value of -32768 (8000 hex) is added to the error code. For example, the error code 5 will be displayed as -32763.

8.2.2 I/O Fault Table

The I/O fault table displays I/O faults such as circuit faults, address conflicts, forced circuits, I/O module addition/loss faults and I/O bus faults.

The fault table displays a maximum of 64 faults. When the fault table is full, it displays the earliest 32 faults (33—64) and the last 32 faults (1—32). When another fault is received, fault 32 is shoved out of the table. In this way, the first 32 faults are preserved for the user to view.



The I/O fault table provides the following information for each fault:

Location Identifies the location of the fault by rack.slot.device.rack.slot.subslot. (For an RXi Controller, the rack.slot designation is always 0.0.)

CIRC No. When applicable, identifies the specific I/O point on the module.

Variable Name If the fault is on a point that is mapped to an I/O variable, and the variable is set to publish (either internal or external), the I/O fault table displays the variable name. Unpublished I/O variables will not be displayed in this field.

Ref. Address If the fault is on a point that is mapped to a reference address, this field identifies the I/O memory type and location (offset) that corresponds to the point experiencing the fault.

Note: The Reference Address field displays 16 bits and %W memory has a 32-bit range. Addresses in %W are displayed correctly for offsets in the 16-bit range (≤65,535). For %W offsets greater than 16-bits, the I/O Fault Table displays a blank reference address.

Fault Category Specifies a general classification of the fault.

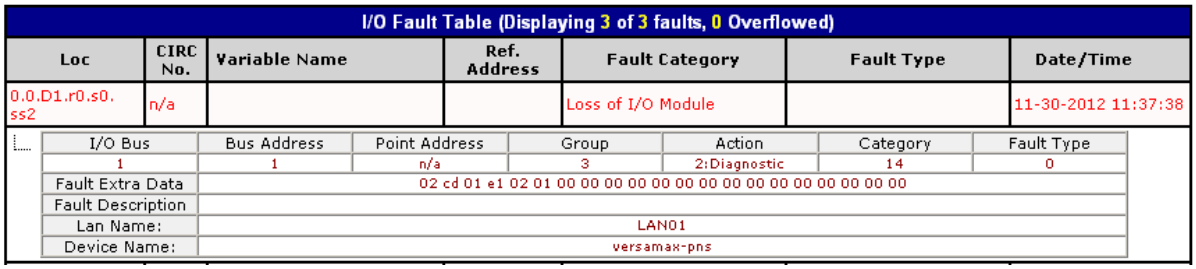
Fault Type Consists of subcategories under certain fault categories. Set to zero when not applicable to the category.

Date/Time The date and time the fault occurred based on the CPU clock.

Details To view detailed information, click the fault entry. See “Viewing I/O Fault Details” for more information.

8.2.2.1 Viewing I/O Fault Details

To see I/O fault details, click the fault entry. The detailed information box for the fault appears. (To close this box, click the fault.)



I/O Fault Table (Displaying 3 of 3 faults, 0 Overflowed)

Loc	CIRC No.	Variable Name	Ref. Address	Fault Category	Fault Type	Date/Time
0.0.D1.r0.s0.ss2	n/a			Loss of I/O Module		11-30-2012 11:37:38

I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type
1	1	n/a	3	2:Diagnostic	14	0

Fault Extra Data	02 cd 01 e1 02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Fault Description	
Lan Name:	LAN01
Device Name:	versamax-pns

The detailed information for I/O faults includes:

I/O Bus This number is always 1.

Bus Address The serial bus address of the device that reported or has the fault.

Point Address Identifies the point on the I/O device that has the fault when the fault is a point-type fault.

Group Fault group is the highest classification of a fault. It identifies the general category of the fault.

Action Fatal, Diagnostic, or Informational. For definitions of these actions, refer to page 137.

Category Identifies the category of the fault.

Fault Type Identifies the fault type by number. Set to zero when not applicable to the category.

Fault Extra Data Provides additional information for diagnostics by Technical Support engineers. Explanations of this information are provided as appropriate for specific faults in “I/O Fault Descriptions and Corrective Actions” on page 150.

Fault Description Provides a specific fault code when the I/O fault category is a circuit fault (discrete circuit fault, analog circuit fault, low-level analog fault) or module fault. It is set to zero for other fault categories.

8.3 System Handling of Faults

The system fault references listed below can be used to identify the specific type of fault that has occurred. (A complete list of system status references is provided in 7.7, System Status References.)

8.3.1 System Fault References

System Fault Reference	Address	Description
#ANY_FLT	%SC0009	Any new fault in either table since the last power-up or clearing of the fault tables.
#SY_FLT	%SC0010	Any new system fault in the controller fault table since the last power-up or clearing of the fault tables.
#IO_FLT	%SC0011	Any new fault in the I/O fault table since the last power-up or clearing of the fault tables.
#SY_PRES	%SC0012	Indicates that there is at least one entry in the Controller Fault table.
#IO_PRES	%SC0013	Indicates that there is at least one entry in the I/O Fault table.
#HRD_FLT	%SC0014	Any hardware fault.
#SFT_FLT	%SC0015	Any software fault.

On power-up, the system fault references are cleared. If a fault occurs, the positive contact transition of any affected reference is turned on the sweep after the fault occurs. The system fault references remain on until both fault tables are cleared or All Memory in the CPU is cleared.

When a system fault reference is set, additional fault references are also set. These other types of faults are listed in “Fault References for Configurable Faults” on page 143 and “Fault References for Non-Configurable Faults” on page 144.

8.3.1.1 Fault References for Configurable Faults

Fault (Default Action)	Address	Description	May Also Be Set
#SBUS_ER (diagnostic)	%SA0032	System bus error.	#HRD_FLT, #SY_PRES, #SY_FLT
#SFT_IOC[1] (diagnostic)	%SA0029	Non-recoverable software error in an I/O Controller (IOC).	#IO_FLT, #IO_PRES, #SFT_FLT
#LOS_RCK[2] (diagnostic)	%SA0012	Loss of rack (loss of power) or missing a configured rack.	#SY_FLT, #SY_PRES, #IO_FLT, #IO_PRES
#LOS_IOC[3] (diagnostic)	%SA0013	Loss of I/O Controller or missing a configured Bus Controller.	#IO_FLT, #IO_PRES
#LOS_IOM (diagnostic)	%SA0014	Loss of I/O module (does not respond), or missing a configured I/O module.	#IO_FLT, #IO_PRES
#LOS_SIO (diagnostic)	%SA0015	Loss of intelligent module (does not respond), or missing a configured module.	#SY_FLT, #SY_PRES
#IOC_FLT (diagnostic)	%SA0022	Non-fatal bus or I/O Controller error, more than 10 bus errors in 10 seconds. (Error rate is configurable.)	#IO_FLT, #IO_PRES
#CFG_MM (fatal)	%SA0009	Configuration mismatch. Wrong module type detected. The CPU does not check the configuration parameter settings for individual modules.	#SY_FLT, #SY_PRES
#OVR_TMP (diagnostic)	%SA0008	CPU temperature has exceeded its normal operating temperature.	#SY_FLT, #SY_PRES

[1] The #SFT_IOC software fault will have the same action as what you set for #LOS_IOC.

[2] When a Loss of Rack or Addition of Rack fault is logged, individual loss or add faults for each module in that rack are usually not generated.

[3] Even if the #LOS_IOC fault is configured as Fatal, the CPU will not go to STOP/FAULT.

Note: If the fault action for a fault logged to the fault table is informational, the configured action is not used. For example, if the logged fault action is informational, but you configure it as fatal, the action is still informational.

8.3.1.2 Fault References for Non-Configurable Faults

Fault	Address	Description	Result
#HRD_CPU (fatal)	%SA0010	CPU hardware fault (such as failed memory device).	Sets #SY_FLT, #SY_PRES, #HRD_FLT
#HRD_SIO (diagnostic)	%SA0027	Non-fatal hardware fault on any module in the system, such as failure of a serial port on a LAN interface module.	Sets #SY_FLT, #SY_PRES, #HRD_FLT
#SFT_SIO (diagnostic)	%SA0031	Non-recoverable software error in a LAN interface module.	Sets #SY_FLT, #SY_PRES, #SFT_FLT
#PB_SUM (fatal)	%SA0001	Program or block checksum failure during power-up or in Run mode.	Sets #SY_FLT, #SY_PRES
#LOW_BAT (diagnostic)	%SA0011	Energy Pack status. For details, see “Battery Status (Group 18)” on page 147.	Sets #SY_FLT, #SY_PRES
#OV_SWP (diagnostic)	%SA0002	Constant sweep time exceeded.	Sets #SY_FLT, #SY_PRES
#SY_FULL #IO_FULL (diagnostic)	%SA0022	Controller fault table full (64 entries). I/O fault table full (64 entries).	Sets #SY_FLT, #SY_PRES, #IO_FLT, #IO_PRES
#APL_FLT (diagnostic)	%SA0003	Application fault.	Sets #SY_FLT, #SY_PRES
#ADD_RCK	%SA0017	Reserved.	Sets #SY_FLT, #SY_PRES
#ADD_IOC (diagnostic)	%SA0018	Extra IOC, or previously faulted I/O Controller is no longer faulted.	Sets #IO_FLT, #IO_PRES
#ADD_IOM (diagnostic)	%SA0019	Extra IO module, or previously faulted I/O module is no longer faulted.	Sets #IO_FLT, #IO_PRES
#ADD_SIO (diagnostic)	%SA0020	New intelligent module is added, or previously faulted module no longer faulted.	Sets #SY_FLT, #SY_PRES
#IOM_FLT (diagnostic)	%SA0023	Point or channel on an I/O module; a partial failure of the module.	Sets #IO_FLT, I#O_PRES
#NO_PROG (information)	%SB0009	No application program is present at power-up. Should only occur the first time the Controller is powered up or if the user memory containing the program fails.	CPU will not go to Run mode; it continues executing Stop mode sweep until a valid program is loaded. This can be a “null” program that does nothing. Sets #SY_FLT and #SY_PRES.
#BAD_RAM (fatal)	%SB0010	Corrupted program memory at power-up. Program could not be read and/or did not pass checksum tests.	Sets #SY_FLT and #SY_PRES.
#WIND_ER (information)	%SB0001	Window completion error. Servicing of Controller Communications or Logic Window was skipped. Occurs in Constant Sweep mode.	Sets #SY_FLT and #SY_PRES.
#BAD_PWD (information)	%SB0011	Change of privilege level request to a protection level was denied; bad password.	Sets #SY_FLT and #SY_PRES.
#NUL_CFG (fatal)	%SB0012	No configuration present upon transition to Run mode. Running without a configuration is equivalent to suspending the I/O scans.	Sets #SY_FLT and #SY_PRES.

Fault	Address	Description	Result
#SFT_CPU (fatal)	%SB0013	CPU software fault. A non-recoverable error has been detected in the CPU. May be caused by Watchdog Timer expiring.	CPU immediately transitions to Stop-Halt mode. The only activity permitted is communication with the programmer. To be cleared, controller power must be cycled. Sets SY_FLT, SY_PRES, and SFT_FLT.
#STOR_ER (fatal)	%SB0014	Download of data to CPU from the programmer failed; some data in CPU may be corrupted.	CPU will not transition to Run mode. This fault is not cleared at power-up, intervention is required to correct it. Sets SY_FLT and SY_PRES.

8.3.2 Using Fault Contacts

Fault (-[F]-) and no-fault (-[NF]-) contacts can be used to detect the presence of I/O faults in the system. These contacts cannot be overridden. The following table shows the state of fault and no-fault contacts.

Condition	[F]	[NF]
Fault Present	ON	OFF
Fault Absent	OFF	ON

An NF contact will be ON (F contact will be OFF) when the referenced I/O point is not faulted, ***or*** the referenced I/O point does not exist in the hardware configuration.

8.3.3 Using Point Faults

Point faults pertain to external I/O faults, although they are also set due to the failure of associated higher-level internal hardware (for example, IOC failure or loss of a rack). To use point faults, they must be enabled in Hardware Configuration on the Memory parameters tab of the CPU.

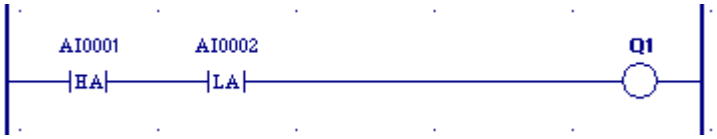
When enabled, a bit for each discrete I/O point and a byte for each analog I/O channel are allocated in CPU memory. The CPU memory used for point faults is included in the total reference table memory size. The FAULT and NOFLT contacts described in “Using Fault Contacts” on page 145 provide access to the point faults.

The full support of point fault contacts depends on the capability of the I/O module. Some modules do not support point fault contacts. The point fault contacts for these modules remain all off, unless a Loss of I/O Module occurs, in which case the CPU turns on all point fault contacts associated with the lost module.

8.3.4 Using Alarm Contacts

High (-[HA]-) and low (-[LA]-) alarm contacts are used to represent the state of the analog input module comparator function. To use alarm contacts, point faults must first be enabled in Hardware Configuration on the Memory parameters tab of the CPU.

The following example logic uses both high and low alarm contacts.



Note: HA and LA contacts do not create an entry in a fault table.

8.4 Controller Fault Details

8.4.1 Controller Fault Groups

Group	Name	Default Fault Action[1]	Configurable	Comments
4	Loss of or Missing Option Module	Diagnostic	No	If one of these faults is logged against the Ethernet interface, contact Technical Support and provide the information contained in the fault entry.
8	Reset of, Addition of, or Extra Option Module	N/A	No	If one of these faults is logged against the Ethernet interface, contact Technical Support and provide the information contained in the fault entry.
11	System Configuration Mismatch	Fatal	No	Occurs when the module occupying a slot is different from that specified in the configuration file. If a system configuration mismatch occurs when the CPU is in Run mode, the fault action will be Diagnostic regardless of the fault configuration. For additional information, see 4.1.4, "Fault Parameters."
12	System Bus Error	Diagnostic	No	Occurs when the CPU encounters a bus error.
13	CPU or PNC hardware failure	N/A	No	Occurs when the CPU detects a hardware failure, such as a memory failure or a communications port failure.
14	Module Hardware Failure	N/A	No	Occurs when the CPU detects a non-fatal hardware failure on any module in the system.
16	Option Module Software Failure	N/A	No	Occurs when: ▪ A non-recoverable software failure occurs on an intelligent option module. ▪ The module type is not a supported type. ▪ The Ethernet Interface logs an event in its Ethernet exception log.

[1] The fault action indicated is not applicable if the fault is displayed as informational. Faults displayed as informational, always behave as informational.

Group	Name	Default Fault Action[1]	Configurable	Comments
17	Program or Block Checksum Failure Group	N/A	No	Occurs when the CPU detects error conditions in program or blocks. It also occurs during Run mode background checking. In all cases, the Fault Extra Data field of the controller fault table record contains the name of the program or block in which the error occurred.
18	Battery Status Group	N/A	No	Occurs when the CPU detects a failed or missing Energy Pack
19	Constant Sweep Time Exceeded	N/A	No	Occurs when the CPU operates in Constant Sweep mode and detects that the sweep has exceeded the constant sweep timer. In the fault extra data, the DWORD at byte offset 8 contains the amount of time that the sweep went beyond the constant sweep time (in microsecond units). Stored in Big Endian format.
20	System Fault Table Full	N/A	No	Occurs when the Controller Fault Table reaches its limit (see page 138).
21	I/O Fault Table Full	N/A	No	Occurs when the I/O Fault Table reaches its maximum configured limit (see page 140). To avoid loss of additional faults, clear the earliest entry from the table.
22	User Application Fault	N/A	No	Occurs when the CPU detects a fault in the user program.
24	CPU Over Temperature	Diagnostic	Yes	The Controller's normal operating temperature exceeded.
129	No User Program on Power-up	N/A	No	Occurs when the CPU powers up with its memory preserved but no user program exists in the CPU. The CPU detects the absence of a user program on power-up; the controller stays in Stop mode.
130	Corrupted User Program on Power-up	N/A	No	Occurs when the CPU detects corrupted user RAM. The CPU will remain in Stop mode.
131	Window Completion Failure	N/A	No	Generated by the pre-logic and end-of-sweep processing software in the CPU. The fault extra data contains the name of the task that was executing when the error occurred.
132	Password Access Failure	N/A	No	Occurs when the CPU receives a request to change to a new privilege level and the password included with the request is not valid for that level.

Group	Name	Default Fault Action[1]	Configurable	Comments
134	Null System Configuration for Run Mode	N/A	No	Occurs when the CPU transitions from Stop to one of the Run modes and a configuration file is not present. The transition to Run is permitted, but no I/O scans occur.
135	CPU or PNC System Software Failure	N/A	No	Generated by the operating software of the CPU. They occur at many different points of system operation. When a fatal fault occurs, the CPU immediately transitions to Stop-Halt. The only activity permitted when the CPU is in this mode is communications with the programmer. The only method of clearing this condition is to cycle power on the controller.
137	Communications Failure During Store	N/A	No	Occurs during the store of programs or blocks and other data to the CPU. The stream of commands and data for storing programs or blocks and data starts with a special start-of-sequence command and terminates with an end-of-sequence command.
140	Non-critical CPU or PNC software event	N/A	No	Used for recording conditions in the system that may provide valuable information to Technical Support.

8.4.2 Controller Fault Descriptions and Corrective Actions

Each fault explanation contains a fault description and instructions to correct the fault. Many fault descriptions have multiple causes. In these cases, the error code and additional fault information are used to distinguish among fault conditions sharing the same fault description.

Group/ Error Code	Description	Cause	Recommended Correction
11-10	Unsupported feature	Byte 8 of the fault extra data contains a reason code indicating what feature is not supported.	▪ Update the module to a revision that supports the feature. ▪ Change the module configuration.
11-11	LAN Softswitch/modem mismatch	Configuration of LAN module does not match modem type or configuration programmed by softswitch utility.	▪ Correct configuration of modem type. ▪ Consult the LAN Interface manual for configuration setup.
11-13	DCD length mismatch	Directed control data lengths do not match.	See Fault Extra Data for DCD length mismatch on page 156.
11-14	LAN duplicate MAC address	This LAN Interface module has the same MAC address as another device on the LAN. The module is off the network.	▪ Change the module's MAC address. ▪ Change the other device's MAC address.
11-15	LAN duplicate MAC address resolved	Previous duplicate MAC address has been resolved. The module is back on the network.	Informational. No correction required.
11-16	LAN MAC address mismatch	MAC address programmed by softswitch utility does not match configuration stored from software.	Change MAC address on softswitch utility or in software.
11-37	Controller reference out of range	A reference on either the trigger, disable, or I/O specification is out of the configured limits.	Modify the incorrect reference to be within range, or increase the configured size of the reference data.
11-38	Bad program specification	The I/O specification of a program is corrupted.	Contact Technical Support.
11-39	Unresolved or disabled interrupt reference	An interrupt trigger reference is either out of range or disabled in the I/O module's configuration.	▪ Remove or correct the interrupt trigger reference. ▪ Update the configuration file to enable this particular interrupt.

Group/ Error Code	Description	Cause	Recommended Correction
11-81	Valid module substitution detected.	A module configured for a PROFINET device does not exactly match what is physically present on the device, but the device indicates it is a valid substitution.	The module should operate properly in the presence of this fault. However, to eliminate the fault, try one or more of the following: Make sure the module is plugged in the correct location and move if necessary. Update the configuration stored to the PNC to exactly match the module present. Replace the module on the device to match the configuration.
11-82	Module configuration mismatch during configuration.	A module configured for a PROFINET device does not match the module that is physically present on the device.	The module will not operate properly, therefore try one or more of the following: Make sure the module is plugged in the correct location and move if necessary. Update the configuration stored to the PNC to equal the actual module present Replace the module on the device to match configuration.
11-83	Module configuration mismatch detected after module hot-insertion.	A module has been hot-inserted in to a PROFINET device, and the physical module does not match the configured module.	Either update the configuration stored to the PNC to equal the actual module present, or replace the module on the device to match configuration.
11-all others	Module and configuration do not match	The module occupying a slot is not of the same type that the configuration indicates.	▪ Replace the module in the slot with the type indicated in the configuration. ▪ Update the configuration.
13-110	Time-of-Day clock not battery-backed	The battery-backed value of the time-of-day clock has been lost.	▪ Replace the RTC battery. Do not remove power from the RXi until replacement is complete. Reset the time-of-day clock using your programming software. ▪
13-168	CPU's critical operating temperature exceeded.	The CPU's critical operating temperature exceeded.	Turn off the RXi Controller to allow heat to disperse and implement measures to regulate ambient operating temperature.
13-h	PNC hardware failure.	The PNC has encountered a hardware related failure. h – Error code that provides more information about what part of the PNC's hardware had failure.	Contact Technical Support.

Group/ Error Code	Description	Cause	Recommended Correction
13-All others	CPU hardware failure	For a memory failure in the CPU (one of the faults reported as a CPU hardware failure), the address of the failure is stored in the first four bytes of the Fault Extra Data.	Replace the module
14-450 through 14-454	LAN interface hardware failure		Contact Technical Support.
14-All others	Module hardware failure	A module hardware failure has been detected.	Replace the affected module.
16-1	Unsupported board type	The board is not one of the supported types.	▪ Upload the configuration file and verify that the software recognizes the board type in the file. If there is an error, correct it, download the corrected configuration file, and retry. ▪ Display the controller fault. Contact Technical Support and provide the information contained in the fault entry.
16-2, 16-3	COMM_REQ frequency too high	COMM_REQs are being sent to a module faster than it can process them.	▪ Change the application program to send COMM_REQs to the module at a slower rate or check the completion status of each COMM_REQ before sending the next.
16-4 through 16-399	Option module software failure	Software failure detected on an option module.	▪ Reload software into the indicated module. ▪ Replace the module.
16-400 and higher	LAN system software fault	The Ethernet interface software has detected an unusual condition and recorded an event in its exception log. The Fault Extra Data contains the corresponding event in the Ethernet exception log, which can be viewed by the Ethernet interface's Station Manager function. The first two digits of Fault Extra Data contain the Event type; the remaining data correspond to the four-digit values for Entry 2 through Entry 6. Some exceptions may also contain optional multi-byte SCode and other data.	For information on interpreting the fault extra data, refer to the *PACSystems TCP/IP Communications Station Manager Manual,* GFK-2225.

Group/ Error Code	Description	Cause	Recommended Correction
17-All error codes	Program or block checksum failure	The CPU has detected a corrupted program or block. The name of the offending program block is contained in the first eight bytes of the Fault Extra Data field.	▪ Clear CPU memory and retry the store. ▪ Examine C application for errors. ▪ Display the controller fault table. Contact Technical Support and provide the information contained in the fault entry.
18-0	Failed battery	Not applicable for the initial release.	Not applicable for the initial release.
18-1	Low battery	Not applicable for the initial release.	Not applicable for the initial release.
19-0	Constant sweep	While operating in Constant Sweep mode, the CPU detected that the sweep has exceeded the constant sweep timer. The fault extra data contains the over sweep time in microseconds in 4 bytes.	▪ Increase constant sweep time. ▪ Remove logic from application program..
20-0	System fault table full	The Controller Fault Table has reached its limit (see page 138).	Clear the controller fault table.
21-0	I/O fault table full	The I/O Fault Table has reached its maximum configured limit (see page 140). To avoid loss of additional faults, clear the earliest entry from the table.	Clear the I/O fault table.
22-2	Software watchdog timer expired	The watchdog timer has expired. When this happens, the CPU stops executing the user program and enters Stop-Halt mode. To recover, cycle power to the Controller. Causes of timer expiration include: Looping, via jump, very long program, etc.	Determine what caused the expiration (logic execution, external event, etc.) and correct. Use the system service function block to restart the watchdog timer.
22-7	Application stack overflow	Block call depth has exceeded the CPU capability.	Increase the program's stack size or adjust application program to reduce nesting.
22-17	Program run time error	A run-time error occurred during execution of a program.	Correct the specific problem in the application.
22-34	Unsupported protocol	Hardware does not support configured protocol.	
22-51	Flash read failed	Possible causes: ▪ Files not in flash. (May be caused by power cycle during flash write.) ▪ Could not read from flash because OEM protection is enabled.	
22-52	Memory reference out of range	A user logic memory reference, which is computed during logic execution, is out of range. Includes indirect references, array element references, and potentially other types of references.	Correct logic or adjust memory size in hardware configuration.
22-53	Divide by zero attempted in user logic.	User logic contained a divide by zero operation. (Applies to ST and FBD logic.)	Correct logic.

Group/ Error Code	Description	Cause	Recommended Correction
22-54	Operand is not byte aligned.	A variable in user logic is not properly byte-aligned for the requested operation.	Correct logic or adjust memory size in hardware configuration.
22-59	PSB called by a block whose %L or %P memory is not large enough to accommodate this reference	Parameterized blocks do not have their own %L data, but instead inherit the %L data of their calling blocks. If %L references are used within a parameterized block and the block is called by _MAIN, %L references are inherited from the %P references wherever encountered in the parameterized block (for example, %L0005 = %P0005). For a discussion of the use of local data with parameterized blocks, refer to 6.1.5.2, “Parameterized Blocks.”	Determine which block called the parameterized subroutine block and increase the size of %L or %P memory allocated to the calling block. (To do this, change the Extra Local Words setting in the block’s Properties.) For additional details, see page 156.
24-1	Overtemperature failure.	CPU’s normal operating temperature exceeded.	Turn off RXi to allow heat to disperse and implement measures to regulate ambient operating temperature.
129-0	No user program on powerup	The RXi Controller has powered up with its memory preserved but no user program exists in the CPU. The CPU detects the absence of a user program on power-up; the controller stays in Stop mode.	Download an application program before attempting to go to Run mode.
130-1	Corrupted user RAM on powerup	The CPU detected corrupted user RAM on power-up.	▪ Cycle power. ▪ Examine any C applications for errors. ▪ Replace the RXi Controller.
130-7	User memory not preserved over power cycle	This is expected operation for the initial release.	This is expected operation for the initial release.
131-0	Window completion failure	While operating in Constant Sweep mode, the constant sweep time was exceeded before the programmer window had a chance to begin executing.	Increase the constant sweep timer value.
131-1	Logic window skipped	The logic window was skipped due to lack of time to execute.	▪ Increase base cycle time. ▪ Reduce Communications Window time.
132-0	Password access failure	The CPU has received a request to change to a new privilege level and the password included with the request was not valid for that level.	Retry the request with the correct password.
134-0	System configuration for run mode	The CPU transitioned from Stop to one of the Run modes and a configuration file was not present. The transition to Run is permitted, but no I/O scans occur.	Download a configuration file.
135-90	User shut down requested	The SVC_REQ #13 (User Shut Down) instruction has executed in the application program.	None required. Information-only alarm.
135-216	Processor exception trap	The processor has detected an error condition while executing an instruction. The CPU was placed into Stop-Halt mode.	Cycle power to clear the Stop-Halt condition.

Group/ Error Code	Description	Cause	Recommended Correction
135-218	Critical overtemperature failure	CPU's critical operating temperature exceeded.	Turn off RXi to allow heat to disperse and implement measures to regulate ambient operating temperature.
135-All others	Critical fatal error.	A critical fatal error has occurred during normal Controller operation from which the Controller cannot recover.	Display the controller fault table. Contact Technical Support and provide the information contained in the fault entry. For details, see page 156.
137-0	Communications failure during store	Communications with the programming device performing the store was interrupted another failure that terminates the store occurred. As long as this fault is present in the system, the controller will not transition to Run mode. This fault is *not* automatically cleared on power-up; you must specifically clear the condition.	Clear the fault and retry the download of the program or configuration file.
137-1	Communications lost during run mode store	Communications or power was lost during a Run Mode Store. The new program or block was not activated and was deleted.	Perform the Run Mode Store again.
137-2	Communications lost during cleanup for run mode store	Communications was lost, or power was lost during the cleanup of old programs or blocks during a Run Mode Store. The new program or block is installed, and the remaining programs and blocks were cleaned up.	None required. This fault is informational.
137-3	Power lost during a run mode store	Power was lost in the middle of a Run Mode Store.	Delete and restore the program. This error is fatal.
140-1 through 140-30	Events during power-up	Records conditions that may provide useful information to Technical Support.	No corrective action is required unless this fault occurs with other specific faults.
140-31 through 140-52	Miscellaneous internal system events	Records conditions that may provide useful information to Technical Support.	No corrective action is required unless the fault occurs with other specific faults.
140-53	Access control fault	If data access is prevented because of the Enhanced Security settings, the Controller logs a fault into the fault table. This fault can be used to help diagnose access problems. To prevent overflowing the fault table, only one fault is logged until the fault table is cleared.	No corrective action is required unless this fault occurs with other specific faults. See page 157 for additional information.
140-54 and greater	Miscellaneous internal system events	Records non-critical conditions that may provide useful information to Technical Support.	This fault is informational. No corrective action is required unless this fault occurs with other specific faults.

Controller Fault Notes

This section contains additional information about specific faults listed in 8.4.2 Controller Fault Descriptions and Corrective Actions.

8.4.2.1 Error code 11-13, DCD Length Mismatch Fault Extra Data

Byte	Value
[0]	FF
[1]	Bus address
[2]	Module's directed data length
[3]	Configured module's directed data length

8.4.2.2 Error Code 18-1, Low Battery Fault

The battery status bits indicate the Energy Pack status.

PLC_BAT (%S0014)	LOW_BAT (%SA0011)	Energy Pack Status
0	0	Does not apply to the initial release.
1	1	Energy Pack not connected or has failed
0	1	Does not apply to the initial release.

8.4.2.3 Error Code 22-59, PSB called by a block whose %L or %P memory is not large enough

The maximum size of %L or %P is 8192 words per block. If your application needs more space, consider changing some %P or %L references to %R, %W, %AI, or %AQ. These changes require a recompilation of the program block and a Stop Mode store to the CPU.

It is possible, by using Online Editing in the programming software to cause a parameterized block to use %L higher than allowed because of the way it inherits data. To correct this condition, delete the %L variables from the logic and then remove the unused variables from the variable list. To implement these changes, you must recompile program block and perform a Stop Mode store to the CPU.

8.4.2.4 Error Code 135-xxx,CPU Internal System Error

Error	Fault Extra Data Value (First Byte)	Description
DEVICE_NOT_AVAILABLE	CF	Specific device is not available in the system.
BAD_DEVICE_DATA	CC	Data stored on device has been corrupted and is no longer reliable. Or, Flash Memory has not been initialized.
DEVICE_RW_ERROR	CB	Error occurred during a read/write of the Flash Memory device.
FLASH_INCOMPAT_ERROR	8E	Data in Flash Memory is incompatible with the CPU firmware release due to the CPU firmware revision numbers, the instruction groups supported, or the CPU model number.
ITEM_NOT_FOUND_ERROR	8D	One or more specified items were not found in Flash Memory.

8.4.2.5 Error code 140-53, Access Control Fault

8.4.2.6 Fault example

Location: 0.8 Date/Time: 07-07-2013 17:06:55.087
Group:140 INFO_CPU_SOFTWR - CPU software event
Error Code:53 Action:1 Task Num:3
Extra Data: 00 fa 02 a5 00 00 00 00 01 1e 06 00 00 00 00 00 00 00 01 00 00 00 00 00

Meaning of this example fault

A 1-bit READ request beginning at %S7 was rejected due to an access violation.

Interpreting the Fault Extra Data

Bytes 1 through 8: Ignored when decoding a security-related fault.

Byte 9: The operation during which the fault occurred.

- 01 (as in the example): Read
- 02: Write

Byte 10: The hexadecimal value (HV) that represents a CPU memory area.

HV	Memory area
08	%R (Register memory)
0A	%AI (Analog input memory)
0C	%AQ (Analog output memory)
10	%I (Discrete input memory)
12	%Q (Discrete output memory)
14	%T (Discrete temporary status memory)
16	%M (Discrete momentary internal memory)
18	%SA (Discrete system memory A)
1A	%SB (Discrete system memory B)
1C	%SC (Discrete system memory C)
1E	%S (Discrete system memory)
38	%G (Genius global memory)
C4	%W (Bulk Memory)

Bytes 11–18: 0-based bit offset of the memory area being accessed. The 8-byte value is encoded in little endian format, meaning that the byte values are reversed. In the example, the value is 0x0000000000000006, which is equal to 1-based bit offset 7.

Bytes 19–22: The length in bits of data requested. In the example, 1 bit was requested.

Bytes 23–24: Ignored when decoding a security-related fault.

8.5 I/O Fault Details

The I/O fault table reports the following data about faults:

- Fault Group
- Fault Action
- Fault category
- Fault type
- Fault description

All faults have a fault category, but a fault type and fault group may not be listed for every fault. To view the detailed information pertaining to a fault, click the fault entry in the I/O Fault Table.

An I/O fault table entry contains up to 21 bytes of I/O fault extra data that contains additional information related to the fault. Not all entries contain I/O fault extra data.

Note: The model number mismatch and I/O type mismatch faults are reported in the controller fault table under the System Configuration Mismatch group. They are not reported in the I/O fault table.

8.5.1 I/O Fault Groups

Group Number	Group Name	Default Fault Action[2]	Configurable
2	Loss of or Missing IOC	Diagnostic	No
3	Loss of or Missing I/O module	Diagnostic	No
6	Addition or Reset of, or Extra IOC	N/A	No
7	Addition of or Extra I/O module	N/A	No
9	IOC or I/O Bus Fault	Diagnostic	Yes
10	I/O Module Fault	N/A	No
15	IOC Software Failure	Same As Group 2 [3]	No
16	Module Software Failure	N/A	No
28	PROFINET alarms	Diagnostic	No

[2] The fault action indicated is not applicable if the fault is displayed as informational. Faults displayed as informational, always behave as informational.

[3] The fault action for the IOC Software Failure group 15 always matches the action used by the Loss of or Missing IOC group 2. If the Loss of or Missing IOC group is configured, the IOC Software Failure group is also configured to take the same fault action.

8.5.2 I/O Fault Categories

Category	Fault Type	Fault Description	Fault Extra Data
Circuit Fault (1)	Discrete Fault (1)	Loss of User Side Power (01 hex)	Circuit Configuration
		Short Circuit in User Wiring (02 hex)	Circuit Configuration
		Sustained Overcurrent (04 hex)	Circuit Configuration
		Low or No Current Flow (08 hex)	Circuit Configuration
		Switch Temperature Too High (10 hex)	Circuit Configuration
		Switch Failure (20 hex)	Circuit Configuration
		Point Fault (83 hex)	Circuit Configuration
		Output Fuse Blown (84 hex)	Circuit Configuration
	Analog Fault (2)	Input Channel Low Alarm (01 hex)	Circuit Configuration
		Input Channel High Alarm (02 hex)	Circuit Configuration
		Input Channel Under Range (04 hex)	Circuit Configuration
		Input Channel Over Range (08 hex)	Circuit Configuration
		Input Channel Open Wire (10 hex)	Circuit Configuration
		Over Range or Open Wire (18 hex)	Circuit Configuration
		Output Channel Under Range (20 hex)	Circuit Configuration
		Output Channel Over Range (40 hex)	Circuit Configuration
		Expansion Channel Not Responding (80 hex)	Circuit Configuration
		Invalid Data (81 hex)	Circuit Configuration
	Low-Level Analog Fault (4)	Input Channel Low Alarm (01 hex)	Circuit Configuration
		Input Channel High Alarm (02 hex)	Circuit Configuration
		Input Channel Under Range (04 hex)	Circuit Configuration
		Input Channel Over Range (08 hex)	Circuit Configuration
		Input Channel Open Wire (10 hex)	Circuit Configuration
		Wiring Error (20 hex)	Circuit Configuration
		Internal Fault (40 hex)	Circuit Configuration
		Input Channel Shorted (80 hex)	Circuit Configuration
		Invalid Data (81 hex)	Circuit Configuration
Loss of Block (2)	Not Specified (0)	NA	Device Configuration Number of Input Circuits Number of Output Circuits
Addition of Block (3)	NA	NA	Device Configuration Number of Input Circuits Number of Output Circuits
I/O Bus Fault (6)	Bus Fault (1)	NA	NA
	SBA Conflict (3)		
	PROFINET Multiple media redundancy manager faults	Refer to Group 6 faults in “I/O Fault Descriptions and Corrective Actions“	

Category	Fault Type	Fault Description	Fault Extra Data
Addition of IOC (9)	NA	Extra Module (01 hex) Reset Request (02 hex)	NA
Loss of IOC (10)	NA	NA	Timeout Unexpected State Unexpected Mail Status
IOC Software Faults (11)	NA	NA	NA
Loss of I/O Module (14)	0	Loss of sub-module on PROFINET device	NA
	2	PROFINET supervisor controlling a PROFINET device's submodule	
Addition of I/O Module (15)		Refer to Groups 7 and 15 faults in in "I/O Fault Descriptions and Corrective Actions "	NA
Extra I/O Module (16)	NA	NA	NA
Extra Block (17)	NA	NA	NA
IOC Hardware Faults (18)	NA	Refer to Group 9 faults in "I/O Fault Descriptions and Corrective Actions "	NA
Reset of IOC (27)	NA	NA	NA
PROFINET device communications faults (33)	NA	Refer to Group 3 faults "I/O Fault Descriptions and Corrective Actions "	
PROFINET device port faults (35)	NA	Refer to Group 3 faults "I/O Fault Descriptions and Corrective Actions"	
PROFINET network interface added (34)	NA		
PROFINET network port faults (36)	NA	Addition of network port	
PROFINET alarms (36 – 67)	NA	Faults related to PROFINET alarms and diagnostics.	
PROFINET load (69)	NA	PNC has become heavily loaded.	
PROFINET controller no longer heavily loaded (70)	NA		Alarm details
Duplicate IP address detected on the network (71)	NA		
Duplicate IP address resolved (72)	NA		

8.5.3 I/O Fault Descriptions and Corrective Actions

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
2-10	Loss of or missing IO controller	**Note:** This fault is always displayed as Fatal in the I/O Fault Table, regardless of its configured action. There are no fault types or fault descriptions associated with this category. The CPU generates this error when it cannot communicate with an I/O Controller and an entry for the IOC exists in the configuration file. This fault is also logged when an IOC is hot removed (No corrective action necessary in this case).	▪ Verify that the module in the slot/bus address is the correct module. ▪ Review the configuration file and verify that it is correct. ▪ Replace the module. ▪ If fault is not resolved, display the controller fault table. Contact Technical Support, giving them all the information contained in the fault entry.
3-2-0-0	Loss of Device	A configured PROFINET device is no longer present on the network. ***Notes:*** The network connection issue might be in the interconnecting network between the controller and the device, and not necessarily at the device itself In an MRP ring with a large number of clients, storing a configuration that causes all clients to reconfigure (for example, changing the Domain Name) may generate a large number of Loss/Addition of Device faults. This is expected behavior and all devices should automatically return to operational.	If this is unexpected operation, either re-connect missing PROFINET device on the network, or remove the device from the PNC's configuration.
3-14-0-0	Loss of sub-module on PROFINET device.	A configured PROFINET sub-module is no longer present.	If the submodule is missing, then replace the missing submodule on the device. If the submodule is present, check for a malfunction on the submodule (e.g. loss of field power, hardware failure, etc.).
3-14-2-0	PROFINET Device's sub-module under control of PROFINET Supervisor.	A PROFINET Supervisor has taken control of a PROFINET device's sub-module, for which the PNC is currently configured and to which it is connected.	If this is unexpected operation, investigate the reason for Supervisor taking control. Investigate reason for Supervisor taking control.
3-33-0-0	Loss of network interface on PROFINET device.	A configured network interface on the PROFINET device is no longer present.	Repair or replace the missing or malfunctioning network interface on the device.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
3-33-2-0	PROFINET Device's network interface under control of PROFINET Supervisor.	A PROFINET Supervisor has taken control of a PROFINET device's network interface, for which the PNC is currently configured and to which it is connected.	If this is unexpected operation, investigate the reason for Supervisor taking control.
3-35-0-0	Loss of network port on PROFINET device.	A configured network port on the PROFINET device is no longer present.	Repair or replace the missing or malfunctioning network port on the device.
3-35-2-0	PROFINET Device's network port under control of PROFINET Supervisor.	A PROFINET Supervisor has taken control of a PROFINET device's network port, for which the PNC is currently configured and to which it is connected.	If this is unexpected operation, Investigate reason for Supervisor taking control.
6-27	Reset of IOC	The fault category Reset of I/O Controller has no fault types or fault descriptions associated with it. The default fault action for this category is Diagnostic. The CPU generates this message when an I/O Controller is reset.	No corrective action necessary.
7-3-0-0	Addition of Device.	A configured PROFINET device that was previously missing has been reconnected.	None. Note: In an MRP ring with a large number of clients, storing a configuration that causes all clients to reconfigure (for example, changing the Domain Name) may generate a large number of Loss/Addition of Device faults. This is expected behavior and all devices should automatically return to operational.
7-15-0-0	Addition of Submodule	A configured PROFINET submodule that was previously reported lost has just been added to the device.	None.
7-15-1-0	Submodule released by PROFINET IO Supervisor	A configured PROFINET submodule that was previously controlled by a PROFINET IO Supervisor has just been released.	None.
7-16	Extra I/O module	The fault category Extra I/O Module applies to discrete and analog I/O modules. There are no fault types or fault descriptions associated with this category. The CPU generates this error when it detects an I/O module in a slot that the configuration file indicates should be empty.	▪ Remove the module. (It may be in the wrong slot.) ▪ Update and restore the configuration file to include the extra module.
7-34-0-0	Addition of network interface.	A configured PROFINET interface that was previously reported lost has been added to the device.	None.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
7-36-0-0	Addition of network port.	A configured PROFINET port that was previously reported lost has been added to the device.	None.
7-36-1-0	Network port released by PROFINET IO Supervisor.	A configured PROFINET port that was previously controlled by a PROFINET IO Supervisor has been released.	None.
9-6-10-0	Multiple Media Redundancy Managers have been detected on the network.	The PNC is configured as a Media Redundancy Manager and other Media Redundancy Managers are currently present on the network.	Update the PNC configuration to change its MRP role to either Disabled or Client, or remove all of the other devices on the network that are also acting as a Media Redundancy Manager.
9-6-11-0	Multiple Media Redundancy Managers are no longer present on the network.	The PNC is configured as a Media Redundancy Manager and it had detected that multiple Media Redundancy Managers were on the network, but now those other managers are no longer present.	No action necessary. The network is expected to only have a single Media Redundancy Manager present.
9-6-12-0	The Media Redundancy Manager has detected that the network ring has been broken.	The PROFINET MRP network ring is broken.	The MRP Ethernet ring is broken. Possible causes include damaged or disconnected cable or power loss to a node or switch. Locate and repair the cause(s).
9-6-13-0	The Media Redundancy Manager has detected that the network ring has been repaired.	The PROFINET MRP network ring is closed/okay.	No action necessary. This is the normal, expected state of an MRP Ethernet ring.
9-11-3-1	Invalid MAC address detected.	PNC no longer has a valid MAC address.	Contact customer service.
9-11-3-3	Media redundancy configuration error.	The PNC has encountered a problem attempting to configure media redundancy operation.	Contact customer service.
9-11-5-x[4]	Internal runtime error.	PNC has encountered an internal error during its operation.	Contact customer service.
9-18-1-1	PNC exceeded its recommended operating temperature.	The temperature detected by the PNC has exceeded its safe operating temperature.	Reduce the temperature of the environment where the PNC is operating.
9-18-1-2	Watchdog Timeout Error	PNC application code restarted due to a hardware watchdog timeout.	Contact customer service.

[4] X – Type value will be equal to the Channel Error Type field of the PROFINET Alarm.
Y – Description value will be equal to the Extended Channel Error Type field of the PROFINET Alarm.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
9-69-0-0	PNC has become heavily loaded.	The PNC has become heavily loaded with activity (Continued use in this mode may cause degradation of system performance (possibly including delayed IO updates) and potential loss of network communications.	Reduce load on PNC. One or more of the following can be tried to reduce load: • Increase the configured update rate value (i.e. IO received less frequently) of one or more PROFINET Devices configured on the PNC. • Reduce the number of PROFINET Devices configured on the PNC.
9-70-0-0	PNC is no longer heavily loaded.	Load applied to PNC has been reduced to an acceptable level, after it was previously heavily loaded.	None.
9-71-0-0	Duplicate IP address detected on the network.	PNC has detected a duplicate IP address on the network. The location associated with the fault will indicate whether the duplication is with the PNC itself, or with a configured PROFINET Device.	Either remove the device on the network that has the duplicate IP address, or assign a new IP address to either the PNC or the conflicting device. **Note:** The Fault Extra Data displays the MAC addresses of the conflicting devices in the MAC 1 field (bytes 8–13) and the MAC 2 field (bytes 14–19). If the PNC is one of the devices with a conflicting IP address, MAC 1 will be 0. This data is stored in Big Endian (most significant byte in lowest address) format. Also, if the conflict is between PROFINET devices on the network, the DCP Tool can be used to help find the conflict. Sorting by IP address within the tool may make it easier to find the duplicates.
9-72-0-0	Duplicate IP address conflict resolved.	PNC has detected that a previously duplicated IP address conflict has been resolved. The location associated with the fault will indicate whether the duplication was with the PNC itself, or with a configured PROFINET Device.	None.
10-8-0-128	Module fault	An internal failure has been detected in a module.	Replace the affected module.
10-8-0-129	Watchdog timeout	The CPU generates this error when it detects that an input module watchdog timer has expired.	Replace the input module.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
10-8-0-132	Output fuse blown	The CPU generates this error when it detects a blown fuse on an output module.	▪ Determine and repair the cause of the fuse blowing, and replace the fuse. ▪ Replace the module.
10-9-0-1	Extra module	Module present, but not configured.	Update the configuration file or remove the module.
10-9-0-2	Reset request	Module added back after reset request.	Informational. No action necessary. No action is necessary if the faulted module is in a remote rack and is returning due to a remote rack power cycle.
28-37-0-0	Unexpected PROFINET Alarm received.	The PNC has received a PROFINET alarm that is unexpected. Possible causes could include a malformed PROFINET Alarm packet or an Alarm for a PROFINET sub-module that is not configured.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-38-0-0	Manufacturer specific Diagnosis Appears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a manufacturer specific diagnostic condition has been detected on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-39-0-0	Manufacturer specific Diagnosis Disappears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a manufacturer specific diagnostic condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data
28-40-x-0[5]	Channel Diagnosis Appears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data.
28-41-x-0	Channel Diagnosis Disappears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data

[5] X – Type value will be equal to the Channel Error Type field of the PROFINET Alarm.
Y – Description value will be equal to the Extended Channel Error Type field of the PROFINET Alarm.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
28-42-x-y[6]	Channel Diagnosis Appears PROFINET Alarm received (Alarm contains Extended Channel data).	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data. For complex analog modules, the reference address displayed in the fault identifies faults in all channels using the discrete status data of the module's first channel. If the fault is on channel 1, the reference address is bit 1 of the channel 1 status data; if the fault is on channel 2, the reference address is bit 2 of the channel 1 status data, etc.
28-43-x-y	Channel Diagnosis Disappears PROFINET Alarm received (Alarm contains Extended Channel data).	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data.
28-44-x-y	Channel Diagnosis Appears PROFINET Alarm received (Alarm contains Qualified Channel data).	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data.
28-45-x-y	Channel Diagnosis Disappears PROFINET Alarm received (Alarm contains Qualified Channel data).	A PROFINET Alarm has been received indicating that a Channel-level diagnostic condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data.
28-46-0-0	Diagnosis Appears PROFINET Alarm received (Alarm contains only Maintenance status).	A PROFINET Alarm has been received indicating that a diagnostic condition has been detected on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data.
28-47-0-0	Diagnosis Disappears PROFINET Alarm received (Alarm contains only Maintenance status).	A PROFINET Alarm has been received indicating that a diagnostic condition has been resolved on the PROFINET device and maintenance is no longer required.	None. Note: Alarm details are provided in the Fault Extra Data.
28-48-0-0	Diagnosis Appears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a diagnostic condition has been detected on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data.

[6] X – Type value will be equal to the Channel Error Type field of the PROFINET Alarm.
Y – Description value will be equal to the Extended Channel Error Type field of the PROFINET Alarm.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
28-49-0-0	Diagnosis Disappears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a diagnostic condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data
28-50-0-0	PROFINET Status Alarm received.	A PROFINET Alarm has been received indicating a status change on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-51-0-0	PROFINET Update Alarm received.	A PROFINET Alarm has been received indicating a change to an operating parameter has been changed outside of the PNC's control.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-52-0-0	Port Data Change PROFINET Alarm received (Alarm contains manufacturer specific data).	A PROFINET Alarm has been received indicating a Port change on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-53-x-0[7]	Port Data Error Appears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a Port Error has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-54-x-0	Port Data Error Disappears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a Port Error condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data
28-55-x-y	Port Data Error Appears PROFINET Alarm received (Alarm contains Extended Channel Data).	A PROFINET Alarm has been received indicating that a Port Error has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-56-x-y	Port Data Error Disappears PROFINET Alarm received (Alarm contains Extended Channel Data).	A PROFINET Alarm has been received indicating that a Port Error condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data
28-57-x-y	Port Data Error Appears PROFINET Alarm received (Alarm contains Qualified Channel Data).	A PROFINET Alarm has been received indicating that a Port Error has occurred on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-58-x-y[8]	Port Data Error Disappears PROFINET Alarm received (Alarm contains Qualified Channel Data).	A PROFINET Alarm has been received indicating that a Port Error condition has been resolved on the PROFINET device.	None. Note: Alarm details are provided in the Fault Extra Data

[7] X – Type value will be equal to the Channel Error Type field of the PROFINET Alarm.
Y – Description value will be equal to the Extended Channel Error Type field of the PROFINET Alarm.

Group/ Category Type/ Error Code	Description	Cause	Recommended Correction
28-59-0-0	Port Data Change PROFINET Alarm received (Alarm contains only Maintenance status).	A PROFINET Alarm has been received indicating a Port change on the PROFINET device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-60-0-0	Network Component Problem Alarm received (Alarm contains manufacturer specific data).	A PROFINET Alarm has been received indicating that a network component has encountered a problem on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-61-x-0	Network Component Problem Appears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a network component has encountered a problem on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-62-x-0	Network Component Problem Disappears PROFINET Alarm received.	A PROFINET Alarm has been received indicating that a network component problem has been resolved on the PROFINET Device.	None. Note: Alarm details are provided in the Fault Extra Data
28-63-x-y	Network Component Problem Appears PROFINET Alarm received (Alarm contains Extended Channel Data).	A PROFINET Alarm has been received indicating that a network component has encountered a problem on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-64-x-y	Network Component Problem Disappears PROFINET Alarm received (Alarm contains Extended Channel Data).	A PROFINET Alarm has been received indicating that a network component problem has been resolved on the PROFINET Device.	None. Note: Alarm details are provided in the Fault Extra Data
28-65-x-y	Network Component Problem Appears PROFINET Alarm received (Alarm contains Qualified Channel Data).	A PROFINET Alarm has been received indicating that a network component has encountered a problem on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data
28-66-x-y	Network Component Problem Disappears PROFINET Alarm received (Alarm contains Qualified Channel Data).	A PROFINET Alarm has been received indicating that a network component problem has been resolved on the PROFINET Device.	None. Note: Alarm details are provided in the Fault Extra Data
28-67-0-0	Network Component Problem Alarm received (Alarm contains only Maintenance status).	A PROFINET Alarm has been received indicating that a network component has encountered a problem on the PROFINET Device.	Consult Device manufacturer documentation. Note: Alarm details are provided in the Fault Extra Data

[8] X – Type value will be equal to the Channel Error Type field of the PROFINET Alarm.

Y – Description value will be equal to the Extended Channel Error Type field of the PROFINET Alarm.

Chapter 9. Gigabit Ethernet (GbE) Interface Overview and Operation

The embedded RXi Ethernet Interface enables the controller to communicate with other PACSystems equipment and with Series 90 and VersaMax controllers. The Ethernet Interface provides TCP/IP communications with other controllers and computers running the TCP/IP version of the programming software. These communications use the proprietary SRTP protocol over a four-layer TCP/IP (Internet) stack.

The Ethernet Interface has SRTP server capability. As a *server,* the Ethernet Interface responds to requests from devices such as the programming software, a Host computer running an SRTP application, or another controller acting as a client.

The initial release does not support SRTP client channel, Modbus TCP server, or Ethernet Global Data protocols.

9.1 PACSystems GbE Communications Features

- Full controller programming and configuration services with inactivity timeout
- TCP/IP communication services using SRTP (server)
- Modbus TCP Server, supporting Modbus Conformance classes 0, 1, and 2.
- Modbus TCP Client, supporting Modbus Conformance classes 0, 1, and Function Codes 15, 22, 23, and 24 for Conformance class 2.
- Extended connectivity via IEEE 802.3 CSMA/CD 10Mbps and 100Mbps Ethernet LAN port connector.
- Network switch that has Auto negotiate, Sense, Speed, and crossover detection.
- Direct connection to BaseT (twisted pair) network switch, hub, or repeater without an external transceiver.
- Monitor mode station management and diagnostic tools. For information on using these commands, refer to the *Station Manager Manual,* GFK-2225.
- Internet access for firmware updates via a web page through standard web browsers.

9.2 Station Manager

The built-in Station Manager function of the Ethernet Interface provides on-line monitor mode access to the Ethernet Interface over the Ethernet cable.

For remote Station Manager operation over the Ethernet network, the Ethernet interface uses IP addressing. A PACSystems Ethernet Interface cannot send or receive remote Station Manager messages sent to a MAC address.

Refer to the *PACSystems TCP/IP Ethernet Communications Station Manager Manual*, GFK-2225 for complete information on the Station Manager.

9.3 IP Addressing

Each TCP/IP node on a network must have a unique IP address. The TCP/IP Ethernet Interface is such a node, as is a PC running TCP/IP. There may be other nodes on the network that are not involved with communications to the controllers, but regardless of function, each TCP/IP node must have its own IP address. It is the IP address that identifies each node on the IP network (or system of connected networks). The term “host” is often used to identify a node on a network.

9.4 SRTP Server

Remote Series 90 PLCs that use SRTP Channels COMM_REQs expect the CPU to be in slot 1. In order to support communications with Series 90 SRTP clients such as Series 90 PLCs using SRTP Channels, the RXi internally redirects incoming SRTP requests addressed to *rack 0, slot 1* to *rack 0, slot 0*.

9.4.1 SRTP Inactivity Timeout

The Ethernet interface supports inactivity checking on SRTP server connections with the Proficy Machine Edition programmer. With this feature, the Ethernet interface removes an abandoned SRTP server connection and all of its resources when there is no activity on the connection for a specified timeout interval. (For example, when communication with the programmer is lost.) Until the server connection is removed, other programmers cannot switch from Monitor to Programmer mode.

Without the SRTP inactivity timeout, an abandoned SRTP server connection persists until the underlying TCP connection eventually times out (typically 7 minutes). All network PME programmer connections initially use an SRTP inactivity timeout value of 30 seconds PME can override the initial timeout value on a particular server connection. Typically the PME programmer sets the SRTP inactivity timeout to 20 seconds. An inactivity timeout value of zero disables SRTP inactivity timeout checking.

The SRTP server uses an internal inactivity timeout resolution of 5 seconds. This has two effects. First, any non-zero inactivity timeout value is rounded up to the next multiple of 5 seconds. Additionally, the actual SRTP inactivity timeout detection for any individual connection may vary up to an additional 5 seconds. The actual inactivity detection time will never be less than the specified value.

Note: The SRTP inactivity timeout applies only to programmer connections over SRTP. It does not affect HMI or SRTP channels.

9.5 Modbus TCP Server Operation

The Modbus TCP Server supports up to 16 simultaneous connections. These connections are not shared with any other applications. Other TCP-based application protocols such as SRTP Server use a different set of TCP connections.

9.5.1 Modbus Conformance Classes

PACSystems Modbus TCP Server supports Modbus Conformance classes 0, 1, and 2.

The RXi Ethernet interface has been certified by the Modbus TCP Conformance Test Laboratory to be in conformance with *Conformance Test Policy* Version 2.1.

9.5.2 Server Protocol Services

The Modbus TCP Server responds to incoming Request Connection, Terminate Connection and Request Service messages. The client with which the server is interacting should wait for the server's response before issuing the next Request Service, otherwise requests could be lost.

There is no inactivity timeout in the server. If a client opens a connection, that connection stays open until the connection is terminated.

9.5.3 Station Manager Support

The Modbus TCP Server supports the standard Station Manager monitor mode commands: STAT, TALLY, and TRACE. The Modbus TCP Server task letter is "o."

9.5.4 Reference Mapping

The Modbus protocol's reference table definition is different from the internal structure of the PACSystems reference tables. Modbus refers to Holding Register, Input Register, Input Discrete and Coil tables; PACSystems uses Discrete Input (%I), Discrete Output (%Q), Analog Input (%AI), Register (%R), and Word (%W) reference tables for Modbus data. The following table shows how each Modbus table is mapped to the PACSystems reference tables.

Modbus Reference Tables

Modbus File Access (6xxxx)	*Modbus Holding Register Table (4xxxx)*	*Modbus Input Register Table (3xxxx)*	*Modbus Input Discrete Table (1xxxx)*	*Modbus Coil Table (0xxxx)*	*PACSystems Reference Tables*
---	---	---	1 – 32768 (bits)	---	%I1 – 32768 (bits)
---	---	1 – 32640 (16-bit words)	---	---	%AI1 – 32640 (16-bit words)
---	---	---	---	1 – 32768 (bits)	%Q1 – 32768 (bits)
---	1 – 32640 (16-bit words)	---	---	---	%R1 – 32640 (16-bit words)
F1,R1 – F525,R2880 (16-bit words)	---	---	---	---	%W1 – 5,242,880 (16-bit words)

9.5.4.1 Modbus File Access Table

The Modbus File Access table is mapped exclusively to PACSystems %W memory.

Applicable Functions

- Read File Record
- Write File Record

Translating %W Reference Addresses

To find the PACSystems %W memory address equivalent of a Modbus File and Record:

%W = 10,000 (F-1) + R

To find the Modbus File and Record equivalent of a PACSystems %W memory address:

$$File = \frac{W-1}{10{,}000} + 1$$

(Discard any fractional portion; round the result downward to the next integer value).

Record = W – (10,000 (F – 1))

Caution

If you use the Modbus function Write File Record, and specify multiple record sections, the first N-1 sections will be written to the server's Controller reference memory, even if an error prevents the writing of the last section.

9.5.4.2 Modbus Holding Register Table

The Modbus Holding Register table is mapped exclusively to the CPU Register (%R) table.

Applicable Functions

- Read Multiple Registers
- Write Multiple Registers
- Write Single Register
- Mask Write Register
- Read/Write Multiple Registers

9.5.4.3 Modbus Input Register Table

The Modbus Input Register table is mapped exclusively to the CPU Analog Input (%AI) table.

Applicable Functions

- Read Input Registers

9.5.4.4 Modbus Input Discrete Table

The Modbus Input Discrete table is mapped exclusively to the CPU Discrete Input (%I) table.

Applicable Functions

- Read Input Discretes

9.5.4.5 Modbus Coil Table

The Modbus Coil table is mapped exclusively to the CPU Discrete Output (%Q) table.

Applicable Functions

- Read Coils
- Write Coils
- Write Single Coil

9.5.5 Address Configuration

Address mapping is done in the Machine Edition Hardware Configuration of the CPU. The Modbus TCP Server does not use COMM_REQs to configure address mapping.

Each Controller memory area is mapped to an appropriate Modbus address space. On the Settings tab, Modbus Address Space Mapping can be set to Standard Modbus Addressing or Disabled. If Modbus Address Space Mapping is set to Standard, the Modbus TCP Address Map tab displays the standard reference assignments.

Number	Modbus Register	Start Address	End Address	Controller Memory Address	Length
1	0xxxx – Coil Table	1	32768	%Q00001	32768
2	1xxxx Discrete Table	1	32768	%I00001	32768
3	3xxxx Input Registers	1	1028	%AI00001	1028
4	4xxxx – Register Table	1	1024	%R00001	1024
5	6yxxx – Internal Table	0	0	%W0001	0

When Modbus Address Space Mapping is set to Disabled on the Settings tab, the Modbus TCP Address Map tab does not appear.

If the CPU does not receive an address map from PME, the Ethernet interface will respond to Modbus TCP clients with Exception Code 4, Slave Device Failure. This same exception code will also be returned when the Controller's hardware configuration is cleared.

9.5.6 Modbus Function Codes

This section summarizes the mapping of PACSystems reference tables to Modbus addresses by the Modbus function codes supported by the Modbus TCP Server. The mapping shown in this table assumes that the RXi Controller is configured to use its default reference table sizes.

Modbus Function Code	Modbus			RXi Controller	
	Table	Start Address	Length	Start Address	Length
1 Read Coils 5 Write Single Coil 15 Write Multiple Coils	0xxxx	1	32768	%Q00001	32768
2 Read Discrete Inputs	1xxxx	1	32768	%I00001	32768
3 Read Holding Registers 6 Write Single Register 16 Write Multiple Registers 22 Mask Write Register 23 Read/Write Multiple Registers	4xxxx	1	1024	%R00001	1024
4 Read Input Registers	3xxxx	1	64	%AI00001	64
7 Read Exception Status 8 Diagnostics	n/a	N/a	n/a	n/a	n/a
20 Read File Record 21 Write File Record	6yxxxx	1	0	%W00001	0

9.6 Modbus TCP Client (Channels) Operation

Modbus TCP Client allows the PACSystems controller to initiate data transfer with other Modbus TCP server devices on the network. The RXi Controller supports 16 Modbus TCP Client channels.

Modbus TCP channels are set up in the application program. The Modbus TCP Client supports COMM_REQ driven channel commands to open new channels, close existing channels, and transfer data on an existing channel.

This section explains how to program communications over the Ethernet network using Modbus TCP Channel commands. This information applies only to RXi Controllers being used as clients to initiate Modbus TCP communications.

9.6.1.1 Structure of the Communications Request

The Communications Request is made up of the following elements:

- COMM_REQ instruction
- COMM_REQ Command Block
- Channel Command
- Status Data (COMM_REQ Status word, LAN Interface Status and Channel Status bits)
- The logic program controlling execution of the COMM_REQ Function Block

9.6.1.2 COMM_REQ Function Block

The COMM_REQ instruction triggers the execution of the Channel command. In the COMM_REQ function block, you specify the rack and slot location of the Ethernet interface (0 for RXi), a task value (65536 for RXi), and the address of the location in memory that contains the command block. The COMM_REQ function block provides a fault output that indicates certain programming errors.

For details on programing a COMM_REQ instruction, refer to the Proficy Machine Edition online help.

9.6.1.3 COMM_REQ Command Block

The COMM_REQ Command Block structure that contains information about the Channel command to be executed. The Command Block consists of two parts:

Common Area - includes the address of the COMM_REQ Status word (CRS word).

Data Block Area - describes the Channel command to be executed.

When the COMM_REQ function is initiated, the command block is transferred to the Ethernet interface for action.

9.6.1.4 Modbus TCP Channel Commands

The Channel commands are a set of client commands used to communicate with a server. Up to 16 channels (numbered 1–16) can be established. The channel number is specified in the Command Block for the Channel command. The channel can be monitored using the Channel Status bits.

9.6.1.5 Modbus TCP Status Data

Several types of status data are available to the client application program.

LAN Interface Status Bits (LIS Bits): The LIS bits comprise bits 1–16 of the 80-bit status area. The location of this 80-bit status area is assigned using the Configuration software. The LIS bits contain information on the status of the Local Area Network (LAN) and the Ethernet interface.

Channel Status Bits: The Channel Status bits comprise bits 17–48 (32 bits) of the 80-bit status area. When used for Modbus TCP channels, these bits consist of a *connection open* bit and an unused bit, reserved for future use, for each of the 16 channels that can be established. Status bits for unused channels are always set to zero.

The status bits are updated in the CPU once each controller scan by the Ethernet interface. These bits can be used to prevent initiation of a COMM_REQ function when certain errors occur or to signal a problem on an established channel. The starting location of these bits is set up when the module is configured.

The status bits are updated in the CPU once each controller scan by the Ethernet interface. These bits can be used to prevent initiation of a COMM_REQ function when certain errors occur or to signal a problem on an established channel. The starting location of these bits is set up when the module is configured.

For definitions of the LIS and Channel Status bits, refer to 9.8.1.3, “Ethernet Interface Status Bits.”

COMM_REQ Status Word(CRS Word) : The 16-bit CRS word receives the initial status of the communication request. The location of the CRS word is assigned for each COMM_REQ function in the COMM_REQ Command Block.

FT Output of the COMM_REQ Function Block: This output indicates that the RXi CPU detected errors in the COMM_REQ function block and/or Command Block and did not pass the Command Block to the Ethernet interface.

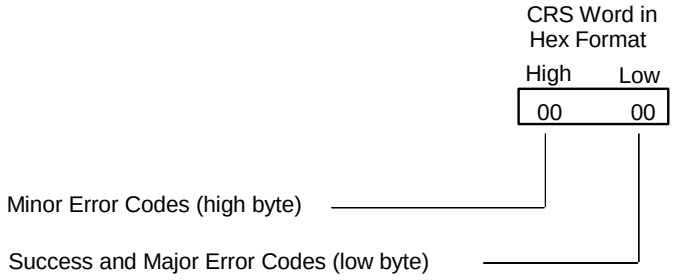
9.6.1.6 Logic Program Execution of the COMM_REQ Function Block

The COMM_REQ must be initiated by a one-shot to prevent the COMM_REQ from being executed repeatedly each CPU scan, which would overrun the capability of the Ethernet interface and possibly require a manual restart. The LAN Interface OK bit should be used as an interlock to prevent execution of the COMM_REQ function when the Ethernet interface is not operational. Following initiation of a COMM_REQ on a channel, no further COMM_REQs should be issued to that channel until the Ethernet interface returns a non-zero CRS word to the program.

9.6.1.7 COMM_REQ Status Word

The COMM_REQ Status word (CRS word) provides detailed information on the status of the COMM_REQ request. The communications status word is not updated in the CPU each scan. It is generally used to determine the *cause* of a communication error after the COMM_REQ function is initiated. The cause is reported in the form of an error code described later in this section. The COMM_REQ Status word (CRS word) is returned from the Ethernet interface to the RXi CPU immediately if the Command Block contains a syntax error or if the command is local. The location of the CRS word is defined in the Command Block for the COMM_REQ function.

The COMM_REQ Status word (CRS word) reports status in the format shown below. The CRS word location is specified in Words 3 and 4 of the Command Block.



The Ethernet Interface reports the status of the COMM_REQ back to the status location. See 9.7.6, “Major and Minor Error Codes in the COMM_REQ Status Word” for COMM_REQ major and minor error codes that may be reported in the CRS words for Modbus TCP commands.

9.6.1.8 FT Output of the COMM_REQ Function Block

This output is set if there is a programming error in the COMM_REQ function block itself, if the rack and slot specified in the COMM_REQ SYSID parameter is incorrect, or if the data block length specified in the Command Block is out of range. This output also may indicate that no more COMM_REQ functions can be initiated in the ladder program until the Ethernet interface has time to process some of the pending COMM_REQ functions.

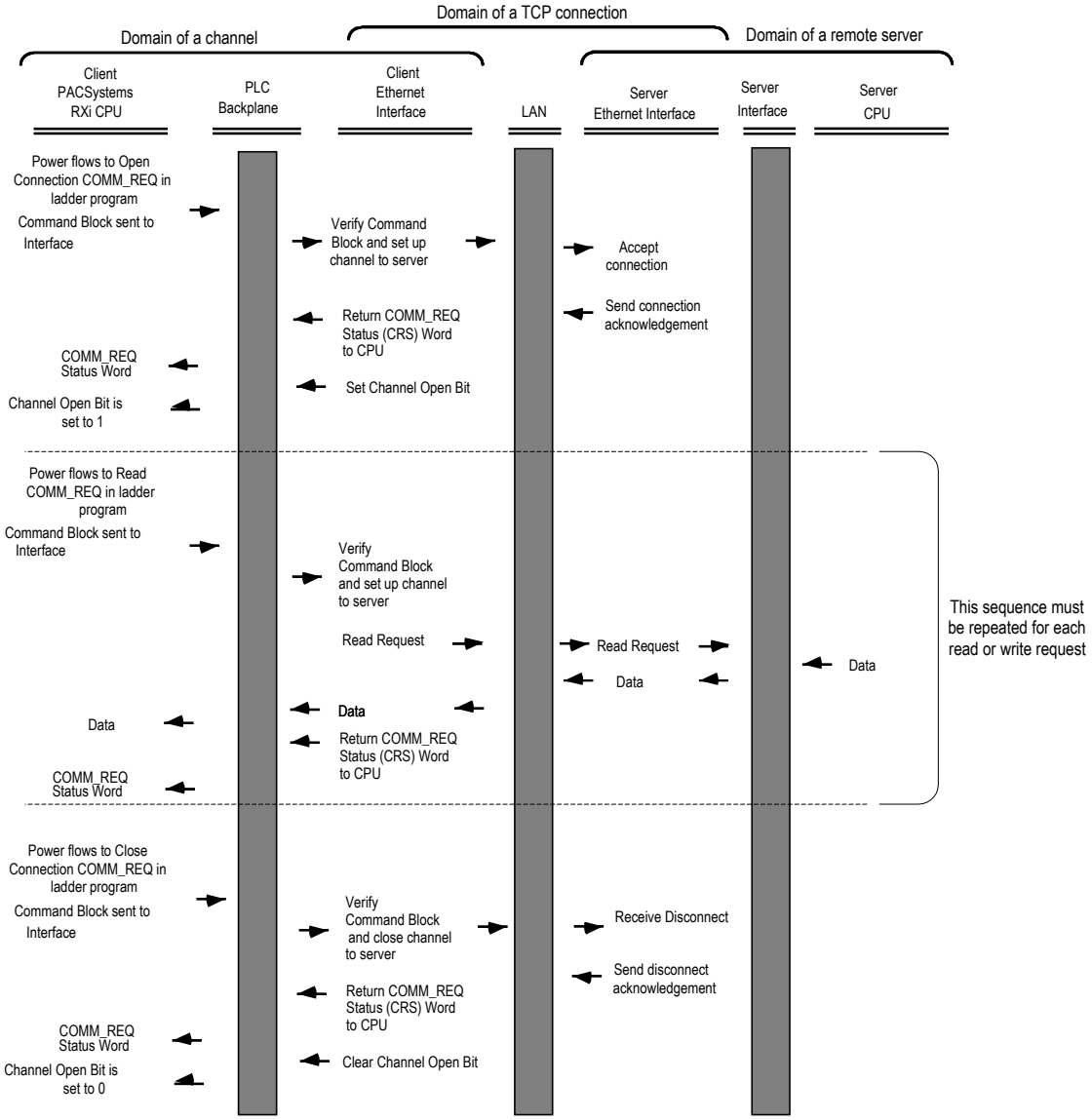
If the FT Output is set, the CPU does not transfer the Command Block to the Ethernet interface. In this case, the other status indicators are not updated for this COMM_REQ.

The FT Output passes power upon the following errors:

- Invalid rack/slot specified. The module at this rack/slot is unable to receive a COMM_REQ.
- Invalid Task ID.
- Invalid Data Block length (zero or greater than 128).
- Too many simultaneous active COMM_REQs (overloading either the CPU or the Ethernet interface).

9.6.2 Operation of the Communications Request

The diagram below shows how Communications Requests are executed to complete a data read from the remote Modbus TCP device. The figure illustrates the successful operation of a data read.


Domain of a channel
Domain of a TCP connection
Domain of a remote server
Client PACSystems RXi CPU
PLC Backplane
Client Ethernet Interface
LAN
Server Ethernet Interface
Server Interface
Server CPU
Power flows to Open Connection COMM_REQ in ladder program
Command Block sent to Interface
Verify Command Block and set up channel to server
Accept connection
Return COMM_REQ Status (CRS) Word to CPU
Send connection acknowledgement
COMM_REQ Status Word
Set Channel Open Bit
Channel Open Bit is set to 1
Power flows to Read COMM_REQ in ladder program
Command Block sent to Interface
Verify Command Block and set up channel to server
Read Request
Read Request
Data
Data
Data
Return COMM_REQ Status (CRS) Word to CPU
COMM_REQ Status Word
This sequence must be repeated for each read or write request
Power flows to Close Connection COMM_REQ in ladder program
Command Block sent to Interface
Verify Command Block and close channel to server
Receive Disconnect
Send disconnect acknowledgement
Return COMM_REQ Status (CRS) Word to CPU
COMM_REQ Status Word
Clear Channel Open Bit
Channel Open Bit is set to 0

1. A Communications Request begins when there is power flow to a COMM_REQ function in the client. The Command Block data is sent from the CPU to the Ethernet interface.
2. The COMM_REQ Status word (CRS word) is returned immediately if the Command Block is invalid. If the syntax is correct, then the CRS word is returned after the transfer of data.

9.6.3 The COMM_REQ Command Block

The Command Block contains the details of a command to be performed by the Interface.

The address in CPU memory of the Command Block is specified by the IN input of the COMM_REQ function block. This address can be any valid address within a word-oriented area of memory. The Command Block is usually set up using either the BLOCK MOVE or the DATA INIT COMM programming instruction. The Command Block has the following structure:

Word 1	Data Block Length (words)
Word 2	WAIT/NOWAIT Flag
Word 3	CRS Word Memory Type
Word 4	CRS Word Address Offset
Word 5	Reserved
Word 6	Reserved
Words 7 and up	Data Block (Channel Command Details)

When entering information for the Command Block, refer to these definitions:

(Word 1) Data Block Length: This is the length in words of the Data Block portion of the Command Block. The Data Block portion starts at Word 7 of the Command Block. The length is measured from the beginning of the Data Block at Word 7, not from the beginning of the Command Block. The correct value for each command, and the associated length of each command, is specified in the next section.

(Word 2) WAIT/NOWAIT Flag: This flag must be set to zero for TCP/IP Ethernet Communications.

COMM_REQ Status Word (CRS): The Ethernet interface updates the CRS word to show success or failure of the command. Command words 3 and 4 specify the CPU memory location of the CRS word.

(Word 3) COMM_REQ Status Word Memory Type: This word specifies the memory type for the CRS word. The memory types are listed in the table below:

Type	*Value (Decimal)*	*Value (Hex)*	*Description*
%R	8	08H	Register memory (word mode)
%AI	10	0AH	Analog input memory (word mode)
%AQ	12	0CH	Analog output memory (word mode)
%I	16 70	10H 46H	Discrete input memory (byte mode) Discrete input memory (bit mode)
%Q	18 72	12H 48H	Discrete output memory (byte mode) Discrete output memory (bit mode)
%T	20 74	14H 4AH	Discrete temporary memory (byte mode) Discrete temporary memory (bit mode)
%M	22 76	16H 4CH	Discrete momentary internal memory (byte mode) Discrete momentary internal memory (bit mode)
%G	56 86	38H 56H	Discrete global data table (byte mode) Discrete global data table (bit mode)
%W	196	C4H	Word memory (word mode limited to %W1 - %W65536)

(Word 4) COMM_REQ Status Word Address Offset: This word contains the offset within the memory type selected. ***The status word address offset is a zero-based number.*** For example, if you want %R1 as the location of the CRS word, you must specify a zero for the offset. The offset for %R100 would be 99 decimal. Note that this is the only zero-based field in the Channel commands.

(Word 5): Reserved. Set to zero.

(Word 6): Reserved. Set to zero.

(Words 7 and up) Data Block: The Data Block defines the Channel command to be performed. For information on how to fill in the Channel command information, see the next section.

9.6.4 Modbus TCP Channel Commands

The RXi Ethernet interface supports the following channel commands:

- Open a Modbus TCP Connection (3000)
- Close a Modbus TCP Connection (3001)
- Read Data from a Modbus Server Device to the controller (3003)
- Write Data from the controller to a Modbus Server Device (3004)
- Read/Write Multiple Registers between controller memory and a Modbus Server Device (3005)
- Mask Write Register Request to a Modbus Server Device (3009)

Note that Modbus TCP channel COMM_REQs do not contain a parameter to configure a timeout value. Enforcing a timeout for a Modbus channel command is at the discretion of the user and must be implemented in the user application.

9.6.4.1 Open a Modbus TCP Client Connection (3000)

The Modbus TCP Ethernet interface transfers data to or from another Modbus TCP device using a *channel*. Up to 16 channels are available for Modbus TCP client communications.

The Open Modbus TCP COMM_REQ requests the communication subsystem to associate a channel with a remote Modbus TCP device. Using the other COMM_REQs defined in this section, the RXi controller can transfer data to and from a remote device.

Once a channel is allocated for Modbus TCP Client communications, the channel remains allocated (i.e. another protocol such as SRTP Channels cannot use the channel). The channel connection is released only when:

- the application program closes the channel
- the channel is automatically closed when the controller transitions to STOP,
- the Ethernet interface is reset or the underlying TCP connection is terminated

The IP address of the remote Modbus TCP device is specified in the Open Modbus TCP COMM_REQ using the standard dotted-decimal format. No other IP address format is accepted.

The COMM_REQ Status Word (CRS) indicates the success or failure of the Open Modbus TCP Client Connection COMM_REQ. If the COMM_REQ requests an invalid channel number or an already allocated channel the COMM_REQ fails and the CRS is set to a non-zero value to identify the failure. See “9.6.1.5, Modbus TCP Status Data” for detailed CRS failure codes.

Command 3000 Example

Establish a channel (Channel 5) to a remote Modbus TCP device at IP address 10.0.0.1. Return the COMM_REQ Status word to %R10.

	Dec	(Hex)	
Word 1	00008	(0008)	Length of Channel command Data Block
Word 2	00000	(0000)	Always 0 (no-wait mode request)
Word 3	00008	(0008)	Memory type of CRS word (%R)
Word 4	00009	(0009)	CRS word address minus 1 (%R10)*
Word 5	00000	(0000)	Reserved
Word 6	00000	(0000)	Reserved
Word 7	03000	(0BB8)	Open Modbus TCP Client Connection
Word 8	00005	(0005)	Channel number (5)
Word 9	00001	(0001)	Remote Device Address Type
Word 10	00004	(0004)	Length of Remote Device Address
Word 11	00010	(0010)	Numeric value of 1st Octet
Word 12	00000	(0000)	Numeric value of 2nd Octet
Word 13	00000	(0000)	Numeric value of 3rd Octet
Word 14	00001	(0001)	Numeric value of 4th Octet

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 is the command id for an Open Modbus TCP Client Connection COMM_REQ. If successful a TCP connection with the specified device is allocated.

(Word 8) Channel Number: Word 8 specifies the channel number to allocate for the Modbus TCP Client connection. Channels 1–16 can be used for Client communications.

(Word 9) Address Type: Word 9 specifies the type of IP Address specified for the remote device. A value of one (1) is required in this word.

(Word 10) Length of IP Address: Word 10 specifies the length of the IP Address. A value of four (4) is required in this word.

(Word 11) IP Address 1st Octet: Word 10 specifies the value of the first octet of the IP Address.

(Word 12) IP Address 2nd Octet: Word 11 specifies the value of the second octet of the IP Address.

(Word 13) IP Address 3rd Octet: Word 12 specifies the value of the third octet of the IP Address.

(Word 14) IP Address 4th Octet: Word 13 specifies the value of the fourth octet of the IP Address.

9.6.4.2 Close a Modbus TCP Client Connection (3001)

The application program closes a Modbus TCP Client Connection by issuing the Close Modbus TCP Client Connection COMM_REQ. The Close COMM_REQ closes the underlying TCP connection and frees the channel for other communication tasks.

An error response is returned if the channel number in the COMM_REQ identifies a non-Modbus TCP Client connection or an inactive channel.

Command 3001 Example

Terminate the Modbus TCP Client connection established on Channel 5. Return the COMM_REQ Status word to %R10.

	Dec	(Hex)	
Word 1	00002	(0002)	Length of Channel command Data Block
Word 2	00000	(0000)	Always 0 (no-wait mode request)
Word 3	00008	(0008)	Memory type of CRS word (%R)
Word 4	00009	(0009)	CRS word address minus 1 (%R10)*
Word 5	00000	(0000)	Reserved
Word 6	00000	(0000)	Reserved
Word 7	03001	(0BB9)	Close Modbus TCP Client Connection
Word 8	00005	(0005)	Channel number (5)

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 requests the Close channel service.

(Word 8) Channel Command Number: Word 8 identifies a channel previously opened with an Open Modbus TCP Client Connection request. If a Close Modbus TCP Client Connection is sent to a channel that is already closed, a success CRS value of 1 will be returned.

9.6.4.3 Read Data from a Modbus TCP Device (3003

The Read Data from a Modbus TCP Device COMM_REQ requests a data transfer from a Modbus TCP device to the controller. The Read Data COMM_REQ must reference an active Modbus TCP channel previously established with the Open Modbus TCP Client Connection COMM_REQ.

Registers, Coils or Exception Status data may be read from the remote Modbus TCP device. The Modbus Function Code specifies the data type. Valid Function Codes for the Read Data COMM_REQ are presented in the following table.

Function Code	Description	Modbus Server Memory Region Accessed*	Data Unit Size	Maximum Data Units
1	Read Coils	Internal Bits or Physical coils	Bit	2000
2	Read Input Discretes	Physical Discrete Inputs	Bit	2000
3	Read Multiple Registers	Internal Registers or Physical Output Registers	Register (16-bit Word)	125
4	Read Input Registers	Physical Input Registers	Register (16-bit Word)	125
7	Read Exception Status	Server Exception Memory	Byte	Not Applicable
24	Read FIFO Queue	Internal Registers or Physical Output Registers	Register (16-bit Word)	32

The table above describes the general Modbus server memory areas. The actual memory accessed is dependent on how the server maps the Modbus memory regions to the server's local memory.

An Address and Length specify the location of the data in the remote device and the number of data units to transfer. The Length is the number of Registers or Coils to transfer. Modbus Function Code 7, Read Exception Status does not require the address as the remote device retrieves the exception status from an internal location.

When transferring data between server bit or coil memory to RXi Controller bit memory, only the number of bits specified is transferred. For example, if the COMM_REQ requests to read 9 coils from the Remote Device and requests to put the data at %M00001 in the Local Controller (using a bit type memory type), %M00001 through %M00009 will be updated with the data from the Remote Device and %M00010 through %M00016 will be unaffected. However, if server bit or coil memory is transferred to the controller byte or word memory, the following rules apply:

1. Transferring discrete data from the Remote Device to Local Controller Word (16-bit) memory: If the number of requested coils is not a multiple of 16, the data is padded with 0s to a 16-bit boundary. For example if the COMM_REQ requests reading 17 coils from the Remote Device and requests to place this data at %R00010, %R00010 (all 16 bits) and bit 0 of %R00011 will be updated with values from the Remote Device and bits 1 through 15 of %R00011 will be set to 0.

2. Transferring discrete data from the Remote Device to Local Controller byte memory (using byte type memory type): If the number of requested coils is not on an 8-bit boundary, the data is padded with 0s to an 8-bit boundary. For example if the COMM_REQ requests 9 coils from the Remote Device and requests to place this data at %M00001, %M00001 through %M00009 will be updated with values from the Remote Device and %M00010 through %M00016 will be set to 0.

Data returned from the remote device is stored in the controller data area specified in the Read Modbus TCP Device COMM_REQ. Data can be stored in any of the controller data areas. Refer to page 185 for the list of data areas and identification codes for the controller. Note that the first item referred to in each data area is item 1 not item 0.

The COMM_REQ Status Word (CRS) indicates the success or failure of the Read Data COMM_REQ. If the COMM_REQ requests an invalid channel number or any other field is invalid the COMM_REQ fails and the CRS is set to a non-zero value to identify the failure. For detailed CRS failure codes, see "9.6.1.5, Modbus TCP Status Data."

Command 3003 Example 1

Read four Input Registers from Input Registers in the remote Modbus TCP device. Store the registers at location %R20. Return the COMM_REQ Status word to %R10.

	Dec	**(Hex)**	
Word 1	00008	(0008)	Length of Channel command Data Block
Word 2	00000	(0000)	Always 0 (no-wait mode request)
Word 3	00008	(0008)	Memory type of CRS word (%R)
Word 4	00009	(0009)	CRS word address minus 1 (%R10) *
Word 5	00000	(0000)	Reserved
Word 6	00000	(0000)	Reserved
Word 7	03003	(0BBB)	Read from a Modbus TCP Device
Word 8	00006	(0006)	Channel number (6)
Word 9	00004	(0004)	Modbus Function Code (Read Input Registers)
Word 10	00008	(0008)	Local Controller Memory Type
Word 11	00020	(0014)	Local Controller Starting Address
Word 12	00200	(00C8)	Address in the Remote Server
Word 13	00004	(0004)	Number of Registers in the Remote Device
Word 14	00001	(0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Read Data from Modbus TCP Device command block.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 4, Read Input Registers.

(Word 10) Local Controller Memory Type: Words 10-11 specify the location in the local controller where the Ethernet interface will store data received from the remote device. Valid values for Word 10 are listed below.

Type	Value (Decimal)	Description
%W	196	Word memory (word mode)
%R	8	Register memory (word mode)
%AI	10	Analog input memory (word mode)
%AQ	12	Analog output memory (word mode)
%I	16 70	Discrete input memory (byte mode) Discrete input memory (bit mode)
%Q	18 72	Discrete output memory (byte mode) Discrete output memory (bit mode)
%T	20 74	Discrete temporary memory (byte mode) Discrete temporary memory (bit mode)
%M	22 76	Discrete momentary internal memory (byte mode) Discrete momentary internal memory (bit mode)
%SA	24 78	Discrete system memory group A (byte mode) Discrete system memory group A (bit mode)
%SB	26 80	Discrete system memory group B (byte mode) Discrete system memory group B (bit mode)
%SC	28 82	Discrete system memory group C (byte mode) Discrete system memory group C (bit mode)
%S *	30 84	Discrete system memory (byte mode) Discrete system memory (bit mode)
%G	56 86	Discrete global data table (byte mode) Discrete global data table (bit mode)

* Read-only memory, cannot be written to.

(Word 11) Local Controller Memory Address: Word 11 determines the starting address in the local controller in which the data from the remote device is to be stored. The value entered is the offset (1-based) from the beginning of CONTROLLER memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the CONTROLLER's memory ranges. Be sure this area is large enough to contain the requested data without overwriting other application data.

(Word 12) Remote Device Address: Word 12 specifies the address in the remote Modbus TCP device. Note: The function code determines the Modbus server address area, Word 12 is the address within this area.

(Word 13) Number Registers in Remote Device: Words 13 specifies the quantity of registers (16bit words) to read from the remote device.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

Command 3003, Example 2

Read nine (9) Input Discretes starting from Discrete input address 5 in the remote Modbus TCP server. Store the registers at location %T3(bit mode). Return the COMM_REQ Status word to %R10.

	Dec	(Hex)	
Word 1	00008	(0008)	Length of Channel command Data Block (8–14 words)
Word 2	00000	(0000)	Always 0 (no-wait mode request)
Word 3	00008	(0008)	Memory type of CRS word (%R)
Word 4	00009	(0009)	CRS word address minus 1 (%R10) *
Word 5	00000	(0000)	Reserved
Word 6	00000	(0000)	Reserved
Word 7	03003	(0BBB)	Read from a Modbus TCP Device
Word 8	00006	(0006)	Channel number (6)
Word 9	00002	(0002)	Modbus Function Code (Read Input Discretes)
Word 10	00074	(004A)	Local Controller Memory Type
Word 11	00003	(0003)	Local Controller Starting Address
Word 12	00005	(0005)	Address in the Remote Device
Word 13	00009	(0009)	Number of Input Discretes to Read from the Remote Device
Word 14	00001	(0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Read Data from Modbus TCP Device command block.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 2, Read Input Discretes.

(Word 10) Local Controller Memory Type: Words 10-11 specify the location in the local controller where the Ethernet interface will store data received from the remote device. Valid values for Word 10 are listed on page 185.

(Word 11) Local Controller Memory Address: Word 11 determines the starting address in the local controller in which the data from the remote device is to be stored. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges. Be sure this area is large enough to contain the requested data without overwriting other application data.

(Word 12) Remote Device Address: Word 12 specifies the address in the remote Modbus TCP device.

(Word 13) Number Registers in Remote Device: Words 13 specifies the quantity of input discretes to read from the remote device.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

Command 3003, Example 3 – Read Exception Status

Read the Exception Status from the remote Modbus TCP server. Store the Exception Data at location %Q4(bit mode). Return the COMM_REQ Status word to %R10.

	Dec (Hex)	
Word 1	00008 (0008)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10) *
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03003 (0BBB)	Read from a Modbus TCP Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00007 (0007)	Modbus Function Code (Read Exception Status)
Word 10	00072 (0048)	Local Controller Memory Type
Word 11	00004 (0004)	Local Controller Starting Address
Word 12	00000 (0000)	Reserved
Word 13	00001 (0001)	Data Size
Word 14	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Read Exception Status from the Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 7, Read Exception Status.

(Word 10) Local Controller Memory Type: Words 10-11 specify the location in the local controller where the Ethernet interface will store data received from the remote device. Valid values for Word 10 are listed on page 185.

(Word 11) Local Controller Memory Address: Word 11 determines the starting address in the local controller in which the data from the remote device is to be stored. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges. Be sure this area is large enough to contain the requested data without overwriting other application data.

(Word 12) Reserved: Word 12 is reserved and must be set to zero.

(Word 13) Data Size: Word 13 is the data size and must be set to 1.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

Command 3003, Example 4 – Read FIFO Queue

Read the FIFO Queue from the remote Modbus TCP server. Store the FIFO Queue Data at location %W1. Return the COMM_REQ Status word to %R10.

	Dec	(Hex)	
Word 1	00008	(0008)	Length of Channel command Data Block
Word 2	00000	(0000)	Always 0 (no-wait mode request)
Word 3	00008	(0008)	Memory type of CRS word (%R)
Word 4	00009	(0009)	CRS word address minus 1 (%R10) *
Word 5	00000	(0000)	Reserved
Word 6	00000	(0000)	Reserved
Word 7	03003	(0BBB)	Read from a Modbus TCP Device
Word 8	00006	(0006)	Channel number (6)
Word 9	00024	(0018)	Modbus Function Code (Read FIFO Queue)
Word 10	00196	(00C4)	Local Controller Memory Type
Word 11	00001	(0001)	Local Controller Starting Address
Word 12	00048	(0030)	FIFO Pointer Address
Word 13	00001	(0001)	Data Size (Unused)
Word 14	00001	(0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Read Exception Status from the Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 24, Read FIFO Queue.

(Word 10) Local Controller Memory Type: Words 10-11 specify the location in the local controller where the Ethernet interface will store data received from the remote device. Valid values for Word 10 are listed on page 185.

(Word 11) Local Controller Memory Address: Word 11 determines the starting address in the local controller in which the data from the remote device is to be stored. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges. Be sure this area is large enough to contain the requested data without overwriting other application data.

(Word 12) FIFO Pointer Address: Word 12 is the FIFO pointer address in the Remote Device.

(Word 13) Data Size: Word 13 is unused because the return data size is dependent on the number of items in the server's FIFO queue when the command is received. Zero (0) through 32 registers can be returned as a result of this function code.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

9.6.4.4 Write Data to a Modbus TCP Device (3004)

The Write Data to a Modbus TCP Device COMM_REQ requests a data transfer from the controller to a Modbus TCP server. The Write Data COMM_REQ must reference an active Modbus TCP channel previously established with the Open Modbus TCP Client Connection COMM_REQ.

Registers or Coils may be written to the remote Modbus TCP device. The Modbus Function Code specifies the data type. Valid Function Codes for the Write Data COMM_REQ are presented in the following table:

Function Code	Description	Modbus Server Memory Region Accessed	Data Unit Size	Maximum Data Units
5	Write Single Coil	Internal Bits or Physical coils	Bit	1
6	Write Single Register	Internal Registers or Physical Output Registers	Register	1
15	Write Multiple Coils	Internal Bits or Physical coils	Bit	1968
16	Write Multiple Registers	Internal Registers or Physical Output Registers	Register	123

An Address Offset and Length specify the location in the Modbus TCP device and the number of data units to transfer. The Address Offset is the offset from the Base Address for that memory region in the server. The Length is the number of Registers or Coils to transfer.

The source for the data written to the Modbus TCP device can be any of the controller data areas (see page 185).

Function Code 5, Write Single Coil, forces a Coil On or Off. To force a coil off, the value zero (0) is used as the COMM_REQ data value. If the controller memory type is a bit type, the remote device coil is set to the same state as the specified controller memory location. If the controller memory type is a byte or word type, a value of zero (0) is used to force a coil off and a value of one (1) is used to force a coil on.

Function Code 15, Write Multiple Coils, forces multiple Coils On or Off. If the controller memory type is a bit type, remote device coils are set to the same state as the corresponding bits in the specified controller memory location. If the controller memory type is byte or word type, the remote device coils follow the state of the packed bits contained in the byte or word memory. For example, if 16 coils are written to a PACSystems Modbus server starting at %Q1 from the client controller memory at %R1 containing a value of 0x1111, the following remote server coils will be set %Q1, %Q5, %Q9 and %Q13 and the following remote server bits will be cleared: %Q2, %Q3, %Q4, %Q6, %Q7, %Q8, %Q10, %Q11, %Q12, %Q14, %Q15, %Q16.

The COMM_REQ Status Word (CRS) indicates the success or failure of the Write Data COMM_REQ. If the COMM_REQ specifies an invalid channel number or any other invalid field the COMM_REQ fails and the CRS is set to a non-zero value to identify the failure. See “9.6.1.5, Modbus TCP Status Data” for detailed CRS failure codes.

Command 3004, Example 1 – Set Single Register

Write one register from %AI10 to register address 200 in the remote Modbus TCP server. Return the COMM_REQ Status word to %R10. Use channel 6, a channel previously opened with the Open Modbus TCP Client Connection COMM_REQ.

	Dec (Hex)	
Word 1	00008 (0008)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10)*
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03004 (0BBC)	Write to a Modbus TCP Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00006 (0006)	Modbus Function Code – Write Single Register
Word 10	00010 (000A)	Local Controller Memory Type
Word 11	00010 (000A)	Local Controller Starting Address
Word 12	00200 (00C8)	Address in the Remote Device
Word 13	00001 (0001)	Number of Registers in the Remote Device
Word 14	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Write Data to remote Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Function Code 6, Write Single Register.

(Word 10) Local Controller Memory Type: Words 10–11 specify the location in the local controller from where the Ethernet interface will get the data to be written to the remote controller. Valid values for Word 10 are listed on page 185.

(Word 11) Local Controller Starting Address: Word 11 determines the starting address in the local controller from which the data is to be written. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges.

(Word 12) Remote Device Address: specifies the destination register in the remote device.

(Word 13) Number Registers in Remote Device: Word 13 specifies the quantity of registers to write to the remote device. For Function Code 6, Write Single Register this must be set to 1.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

Command 3004, Example 2 – Write Single Coil

Set coil 501 ON in the remote Modbus TCP device using the value at %Q4. Return the COMM_REQ Status word to %R10. Use channel 6, a channel previously opened with the Open Modbus TCP Client Connection COMM_REQ.

	Dec (Hex)	
Word 1	00008 (0008)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10)*
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03004 (0BBC)	Write to a Modbus TCP Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00005 (0005)	Modbus Function Code – Write Single Coil
Word 10	00072 (0048)	Local Controller Memory Type
Word 11	00004 (0004)	Local Controller Starting Address
Word 12	00501 (01F5)	Address in the Remote Device
Word 13	00001 (0001)	Number of Coils in the Remote Device.
Word 14	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Write Data to Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 5 Write Single Coil.

(Word 10) Local Controller Memory Type: Words 10–11 specify the location in the local controller from where the Ethernet interface will get the data to be written to the remote controller. Valid values for Word 10 are listed on page 185.

(Word 11) Local Controller Starting Address: Word 11 determines the starting address in the local controller from which the data is to be written. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges.

(Word 12) Remote Device Address: Word 12 specifies the destination coil address in the Modbus TCP device.

(Word 13). Number Coils in Remote Device: Words 13 specifies the quantity of coils to write to the remote device. For Modbus Function Code 5, Write Single Coil, this must be set to 1.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

Command 3004, Example 3 – Set Multiple Registers

Write the four registers from Discrete Input Memory (%I40 to) address 200 in the remote Modbus TCP server. Return the COMM_REQ Status word to %R10. Use channel 6, a channel previously opened with the Open Modbus TCP Client Connection COMM_REQ.

	Dec (Hex)	
Word 1	00008 (0008)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10)*
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03004 (0BBC)	Write to a Modbus TCP Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00016 (0010)	Modbus Function Code – Write Multiple Registers
Word 10	00016 (0010)	Controller Memory Type
Word 11	00040 (0028)	Controller Starting Address
Word 12	00200 (00C8)	Address in the Remote Device
Word 13	00004 (0004)	Number of Registers in the Remote Device
Word 14	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Write Data to Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Modbus Function Code 16, Write Multiple Registers

(Word 10) Local Controller Memory Type: Words 10–11 specify the location in the local controller where the Ethernet interface will get the data to be written to the remote controller. Values for Word 10 are listed on page 185. The value 16 specifies Discrete Input Memory %I (byte mode).

(Word 11) Local Controller Starting Address: Word 11 determines the starting address in the local controller from which the data is to be written. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges.

(Word 12) Remote Device Address: Word 12 specifies the destination register in the remote Modbus TCP device.

(Word 13) Number Registers in Remote Device: Words 13 specifies the quantity of registers to write to the remote device.

(Word 14) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

9.6.4.5 Read/Write Multiple Registers to/from a Modbus Server Device (3005)

The Read/Write Multiple Registers to/from a Modbus Server Device COMM_REQ is used to read and write data between the remote server and the controller with one COMM_REQ operation. Note, the write operation occurs first and the data exchange does not occur coherently (i.e. data can change in the server between the write and read operations).

Command 3005, Example – Read/Write Multiple Register

Write 10 values starting at %R100 in the Local Controller to register address 200 in the remote Modbus TCP server and read 20 values starting from register 300 in the remote Modbus TCP server and write this value to %R300 in the Local Controller. Return the COMM_REQ Status word to %R10. Use channel 6, a channel previously opened with the Open Modbus TCP Client Connection COMM_REQ.

	Dec (Hex)	
Word 1	00014 (000E)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10)*
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03005 (0BBD)	Read/Write Multiple Registers to/from a Modbus TCP Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00023 (0017)	Modbus Function Code – Read/Write Multiple Registers
Word 10	00008 (0008)	Local Controller Memory Type of memory to write with data read from Remote Device
Word 11	00300 (012C)	Local Controller Starting Address (LSW) of memory to write with data read from Remote Device
Word 12	00000 (0000)	Local Controller Starting Address (MSW) of memory to write with data read from Remote Device (normally 0 unless %W is used)
Word 13	00300 (012C)	Address to Read From on Remote Server
Word 14	00020 (0014)	Number of Memory Units to Read from Remote Device (1 to 125)
Word 15	00008 (0008)	Local Controller Memory Type of memory to use for writing to the Remote Device
Word 16	00100 (0064)	Local Controller Starting Address (LSW) of memory to use for writing to the Remote Device
Word 17	00000 (0000)	Local Controller Starting Address (MSW) of memory to use for writing to the Remote Device (normally 0 unless %W is used)
Word 18	00200 (00C8)	Address to Write to on the Remote Server
Word 19	00010 (000A)	Number of Memory Units to Write to the Remote Device (1 to 121)
Word 20	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Read/Write Multiple Register operation on remote Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Function Code 23, Read/Write Multiple Register.

(Word 10) Local Controller Memory Type (Write With Data Read From Server): Words 10–12 specify the location in the local controller where the Ethernet interface will write data received from the remote server. Values for Word 10 are listed on page 185. The value 8 specifies Register Memory %R.

(Word 11) Local Controller Starting Address LSW (Write With Data Read From Server): Word 11 determines the least significant word (LSW) of the starting address in the local controller from which the data is to be written. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 10. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges.

(Word 12) Local Controller Starting Address MSW (Write With Data Read From Server): Word 12 determines the most significant word (MSW) of the starting address in the local controller from which the data is to be written. This value will typically be 0 unless the address is above 65535 for %W memory.

(Word 13) Remote Device Read Address: Word 13 specifies the register(s) to read from the remote Modbus TCP device.

(Word 14) Number Registers to Read From Remote Device: Words 14 specifies the quantity of registers to read from the remote device.

(Word 15) Local Controller Memory Type (Read Data to Write to Server): Words 15–17 specify the location in the local controller where the Ethernet interface will read data to use for writing to the remote server. Values for Word 15 are listed on page 185. The value 8 specifies Register Memory %R.

(Word 16) Local Controller Starting Address LSW (Read Data to Write to Server): Word 16 determines the least significant word (LSW) of the starting address in the local controller from which the data is to be read. The value entered is the offset (1-based) from the beginning of controller memory for the memory type and mode specified in Word 15. This offset will be either in bits, bytes, or words depending on the mode specified. Valid ranges of values depend on the controller's memory ranges.

(Word 17) Local Controller Starting Address MSW (Read Data to Write to Server): Word 17 determines the most significant word (MSW) of the starting address in the local controller from which the data is to be read. This value will typically be 0 unless the address is above 65535 for %W memory.

(Word 18) Remote Device Write Address: Word 18 specifies the register(s) to be written on the remote Modbus TCP device.

(Word 19) Number Registers to Write To Remote Device: Words 19 specifies the quantity of registers to write to the remote device.

(Word 20) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

9.6.4.6 Mask Write Register Request to a Modbus Server Device (3009)

The Mask Write Register Request to a Modbus Server Device COMM_REQ is used to modify the contents of a specified remote device register using a combination of an AND mask, OR mask and the current register's value. This function is used to set or clear individual bits in a register. The register is modified per the following algorithm:

Register value = ((Current register value) AND (And Mask Value)) OR

((OR Mask Value) AND (NOT(And Mask Value)))

Command 3009, Example – Mask Write Register

Modify register at address 200 in the remote Modbus TCP server and clear all bits except bit 0. Return the COMM_REQ Status word to %R10. Use channel 6, a channel previously opened with the Open Modbus TCP Client Connection COMM_REQ.

	Dec (Hex)	
Word 1	00008 (0008)	Length of Channel command Data Block
Word 2	00000 (0000)	Always 0 (no-wait mode request)
Word 3	00008 (0008)	Memory type of CRS word (%R)
Word 4	00009 (0009)	CRS word address minus 1 (%R10)*
Word 5	00000 (0000)	Reserved
Word 6	00000 (0000)	Reserved
Word 7	03009 (0BC1)	Mask Write Register to a Modbus TCP Server Device
Word 8	00006 (0006)	Channel number (6)
Word 9	00022 (0016)	Modbus Function Code – Write Mask Register
Word 10	00200 (00C8)	Address in the Remote Device
Word 11	00001 (0001)	AND Mask
Word 12	00000 (0000)	OR Mask
Word 13	00001 (0001)	Unit Identifier

* Word 4 (CRS word address) is the only zero-based address in the Command Block. Only this value requires subtracting 1 from the intended address.

(Word 7) Channel Command Number: Word 7 identifies the COMM_REQ as a Mask Write Register operation on remote Modbus TCP device.

(Word 8) Channel Number: Word 8 identifies the channel number previously allocated for communication with the remote Modbus TCP server.

(Word 9) Modbus Function Code: Word 9 specifies Function Code 22, Mask Write Register.

(Word 10) Remote Device Address: specifies the destination register in the remote device.

(Word 11) AND Mask: Word 11 specifies the AND mask to be used in the Mask Write operation. For this example, all bits are cleared except bit 0.

(Word 12) OR Mask: Word 12 specifies the OR mask to be used in the Mask Write operation. In this example, no bits are to be set.

(Word 13) Unit Identifier: This field is typically used by Ethernet to Serial bridges to specify the address of a Modbus Slave on a multidrop link. The Modbus TCP Unit Identifier is a special control code used in a Modbus TCP message block.

9.7 Controlling Communications in the Ladder Program

This section provides tips on how to control communications in your ladder program. Only segments of actual ladder logic are included. Topics discussed are:

- Essential Elements of the Ladder Program
- Troubleshooting Your Ladder Program
- Monitoring the Communications Channel

9.7.1 Essential Elements of the Ladder Program

Every ladder program, whether in the developmental phase or the operational phase, should do the following before initiating a COMM_REQ function.

1. Initiate the COMM_REQ function with a one-shot transitional coil. This prevents sending the same COMM_REQ Command Block more than once.
2. Include at least the LAN Interface OK bit in the LAN Interface Status Word as an interlock contact for the COMM_REQ function. You may choose to add more interlocks.
3. Zero the word location you specify for the COMM_REQ Status (CRS) word and the FT outputs of the COMM_REQ function block before the COMM_REQ function is initiated.
4. Move the command code and parameters for the Channel command into the memory location specified by the IN input of the COMM_REQ function block before the COMM_REQ function is initiated.

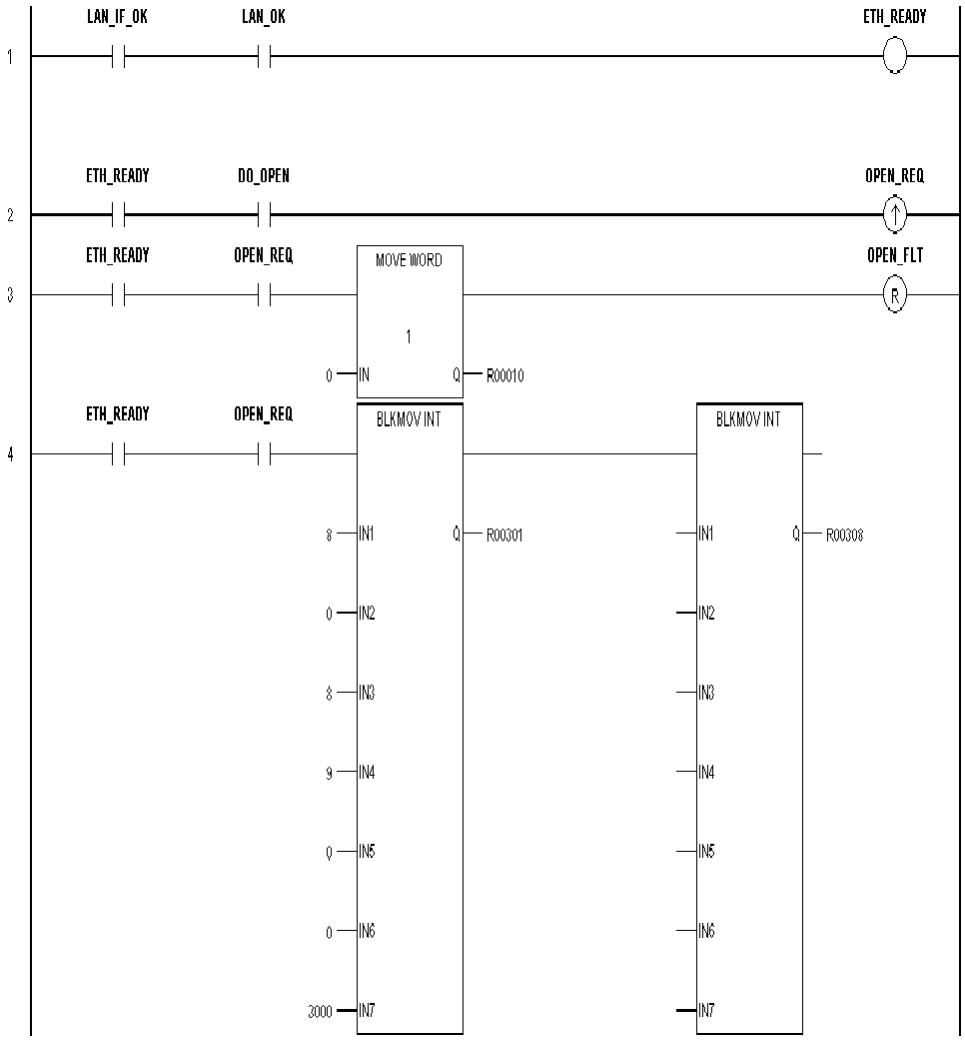
Note: When using a Write Data or Read/Write COMM_REQ, data is not read from the local controller synchronously with execution of the COMM_REQ. A number of CPU sweeps may occur before the data is read. It is recommended that the data not be changed until after the COMM_REQ Status Word indicates completion of the command.

The example ladder program segment starting on the next page illustrates how to incorporate these important points in your program.

9.7.2 COMM_REQ Example

The input values for the Block Move Functions in this example are taken from the Open Modbus TCP Connection (3000), Modbus TCP Read (3003), and Close Modbus TCP Connection (3001) examples in this chapter.

Named variables are used in this example to make the ladder program easier to follow. LAN_IF_OK is bit 16 of the LAN Interface Status bits. LAN_OK is bit 13 of the LAN Interface Status bits.



Rung # 1: Input LAN_IF_OK (bit 16 of the LAN Interface Status bits) monitors the health of the Ethernet interface. Input LAN_OK (bit 13 of the LAN Interface Status bits) monitors the online/offline status of the Ethernet interface. If both bits are set, it is OK to send a COMM_REQ and the ETH_READY coil is ON. ETH_READY is used as an interlock for Rungs 2-16.

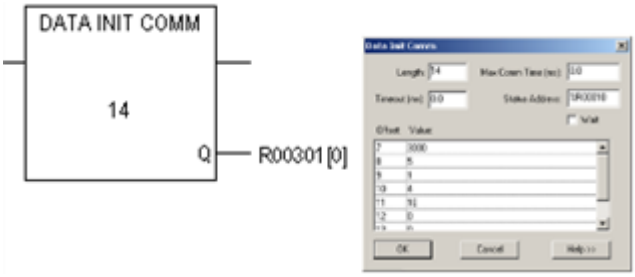
Rung # 2: When ETH_READY is set, Input DO_OPEN triggers OPEN_REQ, which enables execution of the MOVE and COMM_REQ functions for the Open Modbus TCP Connection COMM_REQ. OPEN_REQ is a one-shot (Positive Transition) coil, activating once when both ETH_READY and DO_OPEN have transitioned from OFF to ON.

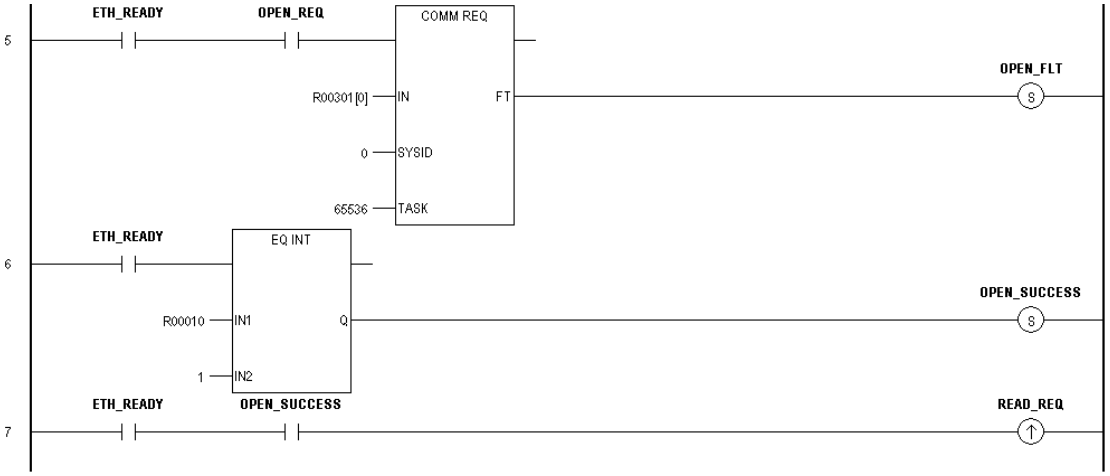
Rung # 3: The MOVE WORD function moves a zero to the CRS word referenced in the Command Block (see rung #4). This clears the CRS word. This rung also resets the OPEN_FLT output coil of the COMM_REQ function block in rung #5.

It is essential that the CRS Status Word and the COMM_REQ fault output coil be cleared each time before initiating a COMM_REQ function.

Rung # 4: The BLKMV INT functions set up the COMM_REQ Command Block contents. When this rung is activated, the constant operands are moved into the memory beginning at the address specified. The constant operands in this example are defined in the Open Modbus TCP Connection Example on page 180.

Note: A single DATA_INIT_COMM function can be used instead of BLKMOV_INT functions to populate the Command Block.



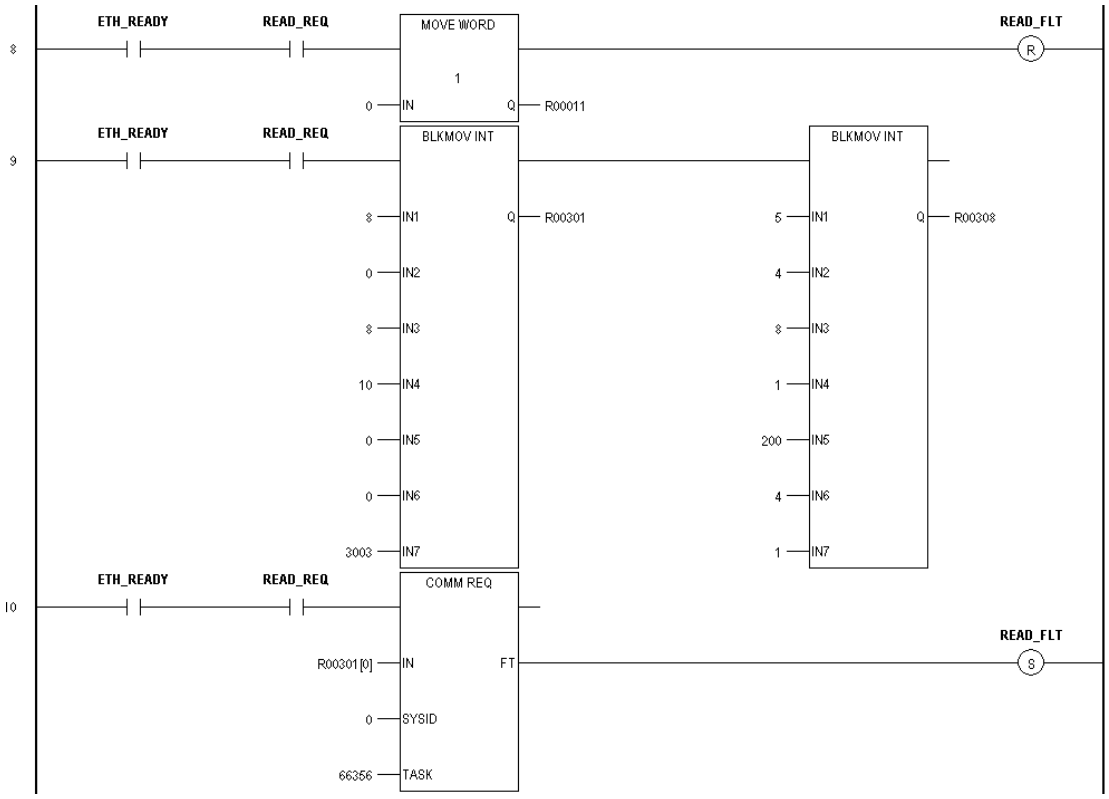


Rung # 5: The COMM_REQ function block has following parameters:

- The IN field points to the starting location of the Command Block parameters (%R00301 in this example).
- The SYSID field defines the target rack and slot of the Ethernet interface to receive the command data. For the RXi Ethernet interface, the value must be 0.
- For the RXi Ethernet interface, TASK must be set to 65536 (0x10000).
- The FT output (energizes the OPEN_FLT coil in this example) is turned ON (set to 1) if there were problems that prevented the delivery of the Command Block to the Ethernet interface. In this case, the other status indicators are not updated for this COMM_REQ.

Rung # 6: When ETH_READY is set the CRS word for the Open Modbus TCP Connection COMM_REQ is monitored for a status of 1, indicating that the Open COMM_REQ completed successfully. The CRS word change to 1 sets coil OPEN_SUCCESS.

Rung # 7: When OPEN_SUCCESS is set it triggers READ_REQ, which enables execution of the BLKMOV, MOVE and COMM_REQ functions for the Modbus TCP Read COMM_REQ. READ_REQ is a one-shot (Positive Transition) coil, activating once when OPEN_SUCCESS transitions from OFF to ON.

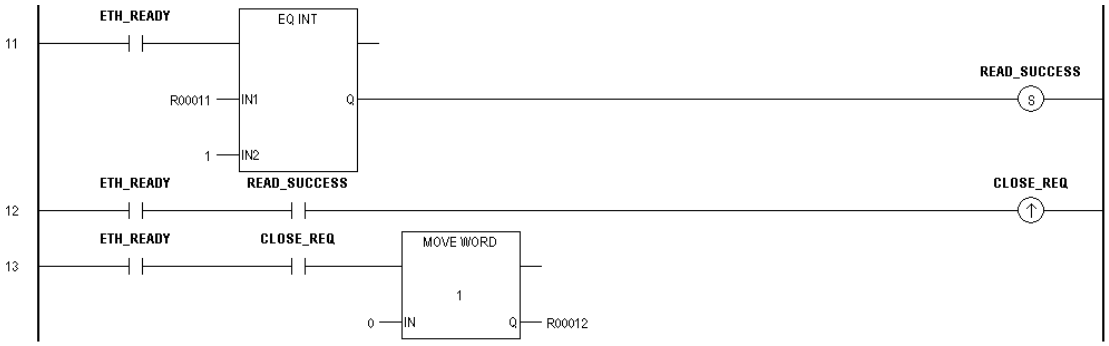


Rung # 8: The MOVE WORD function moves a zero to the CRS word referenced in the Command Block (see rung #9). This clears the CRS word. This rung also resets the READ_FLT output coil of the COMM_REQ function block in rung #10.

Rung # 9: The DATA_INIT_COMM function sets up the COMM_REQ Command Block contents. When this rung is activated, the constant operands are moved into the memory specified in the DATA_INIT_COMM properties. The constant operands in this example are defined in the Modbus TCP Read example on page 183.

Rung # 10: The COMM_REQ function block has the same input parameters has in Rung 5.

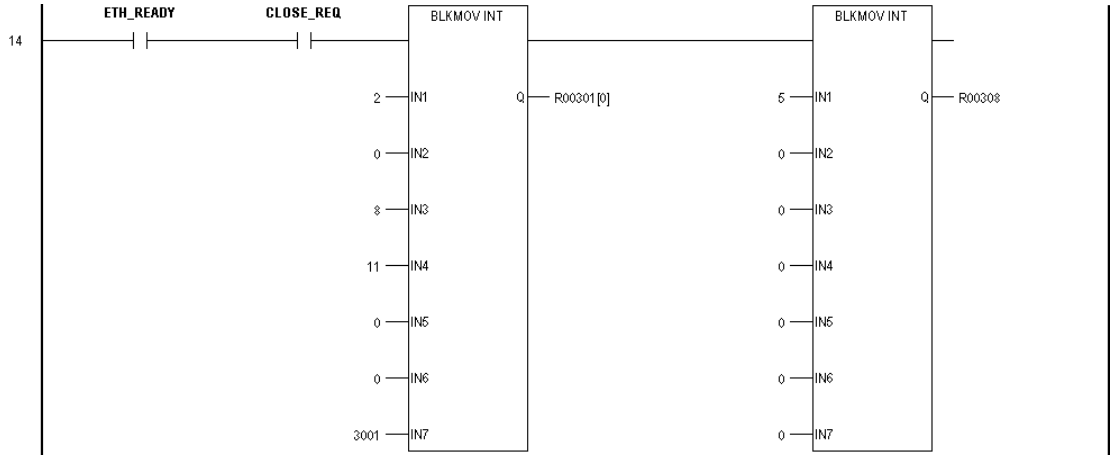
The FT output (energizes the READ_FLT coil in this example) is turned ON (set to 1) if there were problems preventing the delivery of the Command Block to the Ethernet interface. In this case, the other status indicators are not updated for this COMM_REQ.



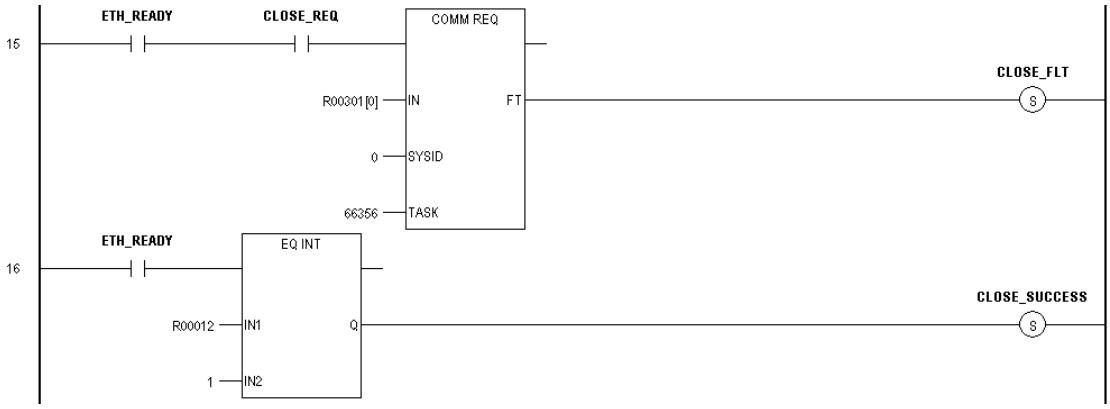
Rung # 11: When ETH_READY is set the CRS word for the Modbus TCP Read COMM_REQ is monitored for a status of 1, indicating that the Read COMM_REQ completed successfully. The CRS word change to 1 sets coil READ_SUCCESS.

Rung # 12: When READ_SUCCESS is set it triggers CLOSE_REQ, which enables execution of the BLKMOV, MOVE and COMM_REQ functions for the Close Modbus TCP Connection COMM_REQ. CLOSE_REQ is a one-shot (Positive Transition) coil, activating once when READ_SUCCESS transitions from OFF to ON.

Rung # 13: The MOVE WORD function moves a zero to the CRS word referenced in the Command Block. This rung also resets the CLOSE_FLT output coil of the COMM_REQ function block in rung #15.



Rung # 14: The DATA_INIT_COMM function sets up the COMM_REQ Command Block contents. When this rung is activated, the constant operands are moved into the memory beginning at the address indicated in the instruction. The constant operands in this example are defined in the Close Modbus TCP Connection example on page 182.



Rung # 15: The COMM_REQ Function has the same input parameters as in Rung 5.

The FT output (energizes the CLOSE_FLT coil in this example) is turned ON (set to 1) if there were problems preventing the delivery of the Command Block to the Ethernet interface. In this case, the other status indicators are not updated for this COMM_REQ.

Rung # 16: When ETH_READY is set the CRS word for the Close Modbus TCP Connection COMM_REQ is monitored for a status of 1, indicating that the Close COMM_REQ completed successfully. The CRS word change to 1 sets coil CLOSE_SUCCESS.

9.7.3 Troubleshooting a Ladder Program

There are several forms of status data that can be accessed by the application program. The use of the LAN Interface OK bit in the LAN Interface Status Word was described in the example program. Some status data can be used to troubleshoot a program in its developmental stage. The two primary sources of this data are the FT Output on the COMM_REQ function block and the COMM_REQ Status word (CRS word).

FT Output is ON

If after executing a COMM_REQ Function, the FT Output is ON, there is a programming error in one or more of the following areas.

- Invalid rack/slot specified. The module at this rack/slot is unable to receive a COMM_REQ Command Block.
- Invalid Task ID. TASK must be set to 65536 (0x10000).
- Invalid Data Block length (0 or greater than 128).

COMM_REQ Status Word is Zero (0) and FT Output is OFF

If after executing a COMM_REQ function, the CRS word is zero (0) and the FT Output is OFF, then the Command Block has been sent to the Ethernet interface, but no status has been returned yet. If this condition persists, check the Controller Fault Table for information.

COMM_REQ Status Word is Not One (1)

If after executing a COMM_REQ function, the CRS word is not one (1) indicating success, then there were:

- Errors in the Command Block (the Channel command code or parameters), *or*
- The command parameters were valid but there was an error in completing the request.

If the CRS word does not contain a 1 indicating success, then it contains either a 0 or a code indicating what error occurred.

9.7.4 Monitoring the Communications Channel

The status data can be used to monitor communications and take action after certain events.

9.7.4.1 Monitoring the COMM_REQ Status Word

It is critical to monitor the CRS word for each COMM_REQ function. First, zero the associated CRS word before executing the COMM_REQ function. When the CRS word becomes non-zero, the Ethernet interface has updated it. If the CRS word is updated to a value of 1, the Command Block was processed successfully by the Ethernet interface. If the CRS word has a value other than 1, an error occurred in processing the Command Block.

Do not use data received from a server until the CRS word for that channel is 1. In addition, do not initiate additional commands to a channel until the CRS word has been updated. The exception to this rule is when you want to terminate a command by using the Close Modbus TCP Connection command.

9.7.4.2 Monitoring the Channel Open Bit

This bit is 1 when a Channel has successfully established a connection with a remote server, and is 0 when a Channel has been closed.. The Channel Open Bit is meaningful when the CPU is in Run mode and the particular channel is being used by Modbus TCP. The Channel Open Bit is set at the same time the successful status is returned to the CRS word for the Open Modbus TCP Connection COMM_REQ.

9.7.5 Sequencing Communications Requests

If the Ethernet interface receives Command Blocks from the CPU faster than it can process them, the Ethernet interface will log an exception event 08, Entry 2=0024H and will log the Controller Fault Table entry:

"Backplane Communications with Controller Fault; Lost Request"

Note: Although there is no backplane in the RXi, this terminology is used because the RXi shares code and fault text strings with the RX3i and RX7i controllers, which do have backplanes.

Only one COMM_REQ function per channel can be pending at one time. A COMM_REQ function is pending from the time it is initiated in the ladder program until its CRS word has been updated to a non-zero value by the Ethernet interface.

9.7.5.1 PACSystems Modbus Client Endian Conversion Example

The following example table shows the Endian conversion behavior for the PACSystems Modbus Client:

Memory Location / Type	Memory value example	Transfer Direction	Memory Location / Type	Resulting Value After Transfer	Notes
Client Bit	%M16-%M1 = 0x4321	→	Server Word	%R1 = 0x4321	End-to-end bytes unswapped
Server Bit	%M16-%M1 = 0x4321	→	Client Word	%R1 = 0x4321	**End-to-end bytes unswapped**
Client Word	%R1 = 0x4321	→	Server Bit	%M16-%M1 = 0x4321	End-to-end bytes unswapped
Server Word	%R1 = 0x4321	→	Client Bit	%M16-%M1 = 0x4321	**End-to-end bytes unswapped**

9.7.6 Major and Minor Error Codes in the COMM_REQ Status Word

9.7.6.1 Major Error Codes in the COMM_REQ Status Word

Success or a Major Error Code appears in the low byte of the COMM_REQ Status Word. Hexadecimal values for the low byte are listed below. For many Major Error Codes, additional information appears as a Minor Error Code in the high byte of the COMM_REQ Status Word. Hexadecimal values for the high byte are listed on the following pages.

Error Status (Hexadecimal)	Major Error Description
01H	Successful Completion. (This is the expected completion value in the COMM_REQ Status word.)
02H	Insufficient Privilege at server controller. For a PACSystems or Series 90-70 server controller, the minor error code contains the privilege level required for the service request.
04H	Protocol Sequence Error. The server CPU has received a message that is out of order. *Contact Technical Support for assistance.*
05H	Service Request Error at server controller. The minor error code contains the specific error code. See the following table of Minor Error codes.
06H	Illegal Mailbox Type at server controller. Service request mailbox type is either undefined or unexpected. *Contact Technical Support for assistance.*
07H	The server controller CPU's Service Request Queue is full, usually due to heavy CPU loading. The client should retry later. It is recommended that the client wait a minimum of 10 milliseconds before sending another service request.
0BH	Illegal Service Request. The requested service is either not defined or not supported at the server controller. (This value is returned in lieu of the actual service request error (01H), to avoid confusion with the normal successful COMM_REQ completion.) *Contact Technical Support for assistance.*
11H	SRTP Error Code at server. An error was detected at the SRTP server. See the following table of Minor Error codes.
82H	Insufficient Privilege at client controller. The minor error code contains the privilege level required for the service request.
84H	Protocol Sequence Error. The CPU has received a message that is out of order. *Contact Technical Support for assistance.*
85H	Service Request Error at the client controller. The minor error code contains the specific error code. See the following table of Minor Error codes.
86H	Illegal Mailbox Type. Service request mailbox type is either undefined or unexpected. *Contact Technical Support for assistance.*
87H	The client controller CPU's Service Request Queue is full. The client should retry later. It is recommended that the client wait a minimum of 10 milliseconds before sending another service request.
8BH	Illegal Service Request. The requested service is either not defined or not supported. (This value is returned in lieu of the actual service request error (01H), to avoid confusion with the normal successful COMM_REQ completion.). *Contact Technical Support for assistance.*
90H	Client (Channels) error. See the following table of Minor Error codes.
91H	Modbus TCP error code at server. An error was detected at the Modbus TCP server. See the following table of Minor Error codes.
A0H	Reserved.

9.7.6.2 Minor Error Codes for Major Error Code 05H (at Remote Server Controller) and 85H (at Client Controller)

Error Status (Hexadecimal)		Error Description
Remote Server	Client	
8F05H	8F85H	Session already exists.
8E05H	8E85H	Memory write is prohibited.
9005H	9085H	Invalid controller memory reference range.
9305H	9385H	Text buffer length/count does not agree with request parameters.
C105H	C185H	Invalid block state transition.
C305H	C385H	Text length does not match traffic type.
C605H	C685H	Control Program (CP) tasks exist but requestor not logged into main CP.
C705H	C785H	Passwords are set to inactive and cannot be enabled or disabled.
C805H	C885H	Password(s) already enabled and cannot be forced inactive.
C905H	C985H	Login using non-zero buffer size required for block commands.
CA05H	CA85H	Device is write-protected.
CB05H	CB85H	A comm or write verify error occurred during save or restore.
CC05H	CC85H	Data stored on device has been corrupted and is no longer reliable.
CD05H	CD85H	Attempt was made to read a device but no data has been stored on it.
CE05H	CE85H	Specified device has insufficient memory to handle request.
CF05H	CF85H	Specified device is not available in the system (not present).
D105H	D185H	Packet size or total program size does not match input.
D205H	D285H	Invalid write mode parameter.
D505H	D585H	Invalid block name specified.
D605H	D685H	Total datagram connection memory exceeded.
D705H	D785H	Invalid datagram type specified.
D805H	D885H	Point length not allowed.
D905H	D985H	Transfer type invalid for this Memory Type selector.
DA05H	DA85H	Null pointer to data in Memory Type selector.
DB05H	DB85H	Invalid Memory Type selector in datagram.
DC05H	DC85H	Unable to find connection address.
DD05H	DD85H	Unable to locate given datagram connection ID.
DE05H	DE85H	Size of datagram connection invalid.
DF05H	DF85H	Invalid datagram connection address.
E005H	E085H	Service in process cannot login.
E405H	E485H	Memory Type for this selector does not exist.
E905H	E985H	Memory Type selector not valid in context.
EA05H	EA85H	Not logged in to process service request.
EE05H	EE85H	Could not return block sizes.
EF05H	EF85H	Programmer is already attached.
F005H	F085H	Request only valid in stop mode.

Error Status (Hexadecimal)		Error Description
Remote Server	Client	
F105H	F185H	Request only valid from programmer.
F205H	F285H	Invalid program cannot log in.
F405H	F485H	Invalid input parameter in request.
F505H	F585H	Invalid password.
F605H	F685H	Invalid sweep state to set.
F705H	F785H	Required to log in to a task for service.
F805H	F885H	Invalid program name referenced.
F905H	F985H	Task address out of range.
FC05H	FC85H	I/O configuration is invalid.
FE05H	FE85H	No privilege for attempted operation.
FF05H	FF85H	Service request has been aborted.

9.7.6.3 Minor Error Codes for Major Error Code 11H (at Remote Server Controller)

Error Status (Hexadecimal)	SRTP Error Description
0111H	Generic SRTP error.
0211H	The controller is inaccessible.
0311H	Reserved.
0411H	Unexpected SRTP version encountered in received message.
0511H	Unrecognized SRTP message received.
0611H	Data present in SRTP message, which should not contain data.
0711H	Generic resource problem detected.
0811H	SRTP message encountered in inappropriate connection state.
0911H	Generic refusal by backplane driver to handle request.
0A11H	Recognized but unsupported SRTP message received.
0B11H	Lost transaction in server.
0C11H	Error sending SRTP PDU to the client controller.
1411H	Unable to allocate a text buffer from dual port memory.
1711H	Invalid text length detected in a mailbox message.
1811H	Invalid number of destinations detected in a mailbox message.
1911H	Invalid source detected in a mailbox message.
1A11H	Invalid slot number detected in a mailbox message.
1B11H	Invalid rack number detected in a mailbox message.
1D11H	Bad text buffer address in dual port memory.
2111H	Unable to find control data required to send a mailbox message to the controller.
2211H	Timed out waiting for availability of mail communications with the controller.
2311H	Invalid task ID detected while attempting to send a mailbox message to the controller.
2411H	Unable to send mailbox message to controller because the mail queue is full.
2611H	Unable to communicate with controller.
2711H	Backplane driver not initialized or unable to acquire a dual port memory semaphore.
2A11H	The backplane driver could not access the controller.
2B11H	Invalid binding on the message sent to the backplane driver.
2C11H	The message could not be sent to its destination because the mailbox was not open.
2D11H	The maximum number of transfers to the destination is already taking place.
2E11H	The maximum number of transfers of this transfer type is already taking place.
2F11H	Cannot obtain a backplane transfer buffer.
3011H	Cannot obtain resources other than backplane transfer buffers.
3111H	Connection ID or block transfer ID is not valid.
3211H	Timed out waiting for controller CPU response.
3311H	The controller CPU aborted the request.
3411H	An invalid message type was specified.
3511H	The specified task is not registered.
3611H	The mailbox offset specified is invalid.
3711H	The backplane task could not be registered because the message response handler was not specified.
3811H	The backplane task could not be registered because the unsolicited mailbox message handler was not specified.
3911H	The backplane task could not be registered because a required parameter was not specified.
3A11H	More than the allowable byte length in a single transfer.
3B11H	Bad sequence number in the request.
3C11H	Invalid command in request.

Error Status (Hexadecimal)	SRTP Error Description
3D11H	Response length does not match length specified in the response qualifier.
3E11H	Request failed because the controller's Service Request Processor is not initialized.
3F11H	Request failed due to an error in the remote device, most likely running out of Dual-Port RAM text buffers.
4011H	Unable to free dual port memory that was allocated for a connection or block transfer area.
4111H	The backplane task could not be registered because the service request handler was not specified.
4211H	No dual port memory was allocated for the connection or block transfer area needed to process the request.
4311H	Failure to register with backplane driver because the requested task is already registered.
4411H	Request failed because an invalid field was identified in the request mailbox qualifier.
E811H	Unable to send request to the controller because an internal message queue is full.
E911H	Unable to send request to the controller because the text buffer type is invalid.
EA11H	Unable to send request to the controller because the mailbox utility function is invalid.
EB11H	Unable to send request to the controller because the mailbox message is not specified.
EC11H	Unable to send request to the controller because the internal message queue is not initialized.
FE11H	Request failed due to mailbox error on remote device. The remote device log will have more information.
2911H	The backplane driver is not initialized.
2A11H	The backplane driver could not access the controller.
2F11H	Request failed due to an invalid parameter detected in the remote device. The remote device log will have more information.
3011H	The specified task is not registered.
3111H	Failure to register with backplane driver because the requested task is already registered.
3211H	Unable to find resource necessary for backplane driver to process a service request.
3311H	Bad sequence number detected in the service request because it is already in use.
3411H	Invalid data detected that prevents backplane driver from completing a request.
3611H	More than the allowable byte length in a single transfer.
4811H	Memory resource problem detected.
4911H	Network buffer resource problem detected.
4C11H	Error detected while attempting to receive mailbox messages from the controller.
4D11H	Timed out waiting to obtain a backplane transfer buffer.
4E11H	Timed out waiting to transfer a mailbox message to the controller.
4F11H	Timed out waiting for controller CPU response.

9.7.6.4 Minor Error Codes for Major Error Code 90H (at Client Controller)

Error Status (Hexadecimal)	Error Description
0190H	Timeout expired before transfer completed; still waiting on transfer.
0290H	Period expired before transfer completed; still waiting on transfer.
8190H	COMM_REQ data block too short for the command.
8290H	COMM_REQ data block too short for server controller node address.
8390H	Invalid server memory type.
8490H	Invalid Program Name.
8590H	Invalid Program Block Name.
8690H	Zero server unit length is not allowed.
8790H	Server unit length is too large.
8890H	Invalid channel number.
8990H	Invalid time unit for period. (Maximum permitted 3965 hours)
8A90H	Period value is too large.
8B90H	Zero server memory starting address is not allowed.
8C90H	Invalid client memory type.
8D90H	Invalid server host address type.
8E90H	Invalid IP address integer value. (Must be 0–255)
8F90H	Invalid IP address class. (Must be valid Class A, B, or C IP address) May also occur if the destination IP address in the COMM_REQ is same as the sender's IP address.
9090H	Insufficient TCP connection resources to do request.
9190H	Zero local starting address is not allowed.
9290H	Address length value invalid. Must be 4 for address type 1.
9390H	COMM_REQ data block too short for Program Block name (including 0 pad).
9490H	COMM_REQ data block too short for Program name (including 0 pad).
9590H	Internal API error. See Controller Fault Table or exception log for details. This problem may occur due to the Ethernet Interface being asked to perform beyond its capacity. Try transferring less data per message or establishing fewer simultaneous connections.
9690H	Underlying TCP connection aborted (reset) by server end point.
9790H	Underlying TCP connection aborted by client end point.
9890H	The remote server has no Service Request Processor.
9A90H	Response to session request did not arrive in proper order.
9B90H	Session denied by server controller.
9C90H	Data response did not arrive in proper order.
9D90H	Data response had unexpected size.
9E90H	Unrecognized COMM_REQ command code.
A190H	Invalid CRS word memory type.
A290H	Failed an attempt to update the CRS word.
A390H	*Reserved.*
A490H	*Reserved.*
A590H	*Reserved.*
A690H	Invalid bit mask.
A790H	Unable to connect to remote device.
A890H	Channel Resources in Use. Try the command again; a resource will become available.
A990H	"Establish Read/Write/Send Info Report Channel" COMM_REQ was received while an Abort was in progress.

Error Status (Hexadecimal)	Error Description
AA90H	An attempt to establish a TCP connection with a Remote Server has failed. Check the following: ■ Make sure the Server is turned on. ■ Make sure cables are connected. ■ If using a switch, make sure the switch is turned on.
AB90H	A COMM_REQ was discarded because the application program issued the COMM_REQ before the COMM_REQ Status Word for the previous COMM_REQ was set.
AC90H	A protocol error occurred while communicating with the local controller.
AD90H	A TCP Timeout occurred while communicating with the Remote Controller.
AE90H	A protocol error occurred while communicating with the local Controller.
B490H	The channel that the application is trying to open is already open.
B590H	The channel the application is trying to access is owned by a different protocol.
B690H	COMM_REQ specified an invalid Modbus function code.
B790H	COMM_REQ specified an invalid Modbus unit ID.
B890H	COMM_REQ specified an invalid number of subrequests.
B990H	A COMM_REQ subrequest specified an invalid record number.
C090H	Reserved.
FF90H	Abort in progress on a channel

9.7.6.5 Minor Error Codes for Major Error Code 91H (at Remote Modbus TCP Server)

The Minor codes for Major Error Code 91H indicate standard Modbus exception codes returned from the remote Modbus TCP server/slave device.

Error Status (Hexadecimal)	Error Description
0191H	Illegal function. The function code received in the query is not an allowable action for the server. (Modbus exception code 01 ILLEGAL FUNCTION)
0291H	Illegal Data Address. The data address received in the query is not an allowable address for the server. The combination of reference number and transfer length is invalid. (Modbus exception code 02 ILLEGAL DATA ADDRESS)
0391H	Illegal Data Value. A value in the query field is not an allowable value for the server. This indicates a fault in the remainder of the request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in the server has an incorrect value. (Modbus exception code 03 ILLEGAL DATA VALUE)
0491H	Slave Device Failure. An unrecoverable error occurred while the server was attempting to perform the requested action. (Modbus exception code 04 SLAVE DEVICE FAILURE)
0591H	Acknowledge. Used for Programmer operations only. Our Modbus TCP server does not support Modbus programmer operations. (Modbus exception code 05 ACKNOWLEDGE)
0691H	Slave Device Busy. The server is unable to accept and process this Modbus request. (Modbus exception code 06 SLAVE DEVICE BUSY)
0791H	Negative Acknowledge. An internal server error occurred while attempting to process a Modbus request. (Modbus exception code 07 NEGATIVE ACKNOWLEDGE)
0891H	Memory Parity Error. (Function codes 20 and 21 only.) The extended file area failed to pass a consistency check. (Modbus exception code 08 MEMORY PARITY ERROR)
0991H	Reserved. (Modbus exception code 09 RESERVED)
0A91H	Gateway Path Unavailable. Gateway was unable to allocate a PATH to process the request. Usually means the gateway is misconfigured or overloaded. (Modbus exception code 10 GATEWAY PATH UNAVAILABLE)
0B91H	Gateway Target No Response. No response was obtained from target device. Usually means that the device is not present on the network. (Modbus exception code 11 GATEWAY TARGET NO RESPONSE)

9.8 Ethernet Interface Diagnostics

9.8.1 Diagnostics Tools

The following tools are available to assist you in diagnosing problems with the Ethernet Interface and the network.

- Controller Fault table
- Ethernet status bits
- Station Manager

9.8.1.1 Controller Fault Table

Use the Controller Fault table to troubleshoot a problem once the Interface is running. It provides a record of exceptions logged by the controller, the Ethernet Interface, and other I/O and communications modules. The Controller Fault table is accessed through the programming software or the IDM.

The table on the next two pages lists Ethernet interface faults and corrective actions.

To access the details of a Controller Fault Table entry, double-click the Fault Table entry and the details are displayed as “fault extra data”. Refer to the online help in the programming software for more information.

An example of the fault extra data is shown below:

```
160006000300050000000000000000000000000000000000
  | |   |   |   |
```

For Ethernet Interfaces the leftmost 14 digits of fault extra data (underlined in the example above) show the corresponding log Events (2 digits) and Entries 2, 3, and 4 (in that order, 4 digits each). The example above is reporting an Event 16, Entry 2=6, Entry 3=3, and Entry 4=5.

For Controller Fault table entries generated by the Ethernet Interface, the Detailed Fault Data for that entry contains the same data as the corresponding event in the Ethernet Interface’s exception log. Refer to GFK-2225, *TCP/IP Ethernet Communications for the PACSystems Station Manager Manual,* for information on how to interpret Ethernet exception log events.

Controller Fault Table Descriptions

Controller Fault	*User Action*
Backplane communications with controller fault; lost request	Check to make sure that the logic application is not sending COMM_REQs faster than the Ethernet Interface can process them. Reduce the rate at which the application is sending COMM_REQs to the Ethernet interface. If problem persists, contact Technical Support.
Mailbox queue full – COMM_REQ aborted	Check to make sure that the logic application is not sending COMM_REQs faster than the Ethernet Interface can process them. Reduce the rate at which the application is sending COMM_REQs to the Ethernet interface. If problem persists, contact Technical Support.
Bad local application request; discarded request	Check for valid COMM_REQ command code. If problem persists, contact Technical Support.
Bad remote application request; discarded request	Try to validate the operation of the remote node. If problem persists, contact Technical Support.

Controller Fault	User Action
Can't locate remote node; discarded request	Error reported when message received where IP/MAC address cannot be resolved. Error may indicate that remote host is not operational on the network. Check that remote host is operational on network and its addresses are correct.
COMM_REQ - Bad task ID programmed	Message from controller for unknown Ethernet Interface task. Check COMM_REQ function block.
COMM_REQ - Wait mode not allowed	Check COMM_REQ to make sure sent in no-wait mode.
Configured gateway address bad; can't talk off local net	Error in configuration. Verify that IP address, Subnetwork Mask, and default Gateway IP address are correct.
Connection to remote node failed; resuming without it	Underlying communications software detects error transferring data; resuming. If persistent error, check connection to LAN and operation of remote node.
LAN controller fault; restart LAN I/F	HW fault, perform a power cycle. If problem persists, contact Technical Support.
LAN controller Tx underflow; attempt recovery	Internal system error. If problem persists, contact Technical Support.
LAN controller under run/overrun; resuming	Internal system error. If problem persists, contact Technical Support.
LAN data memory exhausted - check parameters; resuming	The Ethernet Interface does not have free memory to process communications. If problem persists, contact Technical Support.
LAN duplicate MAC Address; resuming	A frame was received in which the source MAC Address was the same as this station's MAC Address. All stations on a network must have a unique MAC address. Immediately isolate the offending station; it may be necessary to turn it off or disconnect it from the network. This station remains Online unless you intervene to take it Offline.
LAN I/F can't init - check parameters; running soft Sw utl	Internal system error. If problem persists, contact Technical Support.
LAN I/F capacity exceeded; discarded request	Verify that connection limits are not being exceeded.
LAN interface hardware failure; switched off network	Replace the Ethernet Interface.
LAN network problem exists; performance degraded	Excessive backlog of transmission requests due to excessive traffic on the network. For a sustained period the MAC was unable to send frames as quickly as requested. If problem persists, contact Technical Support.
LAN severe network problem; attempting recovery	External condition prevented transmission of frame in specified time. Could be busy network or network problem. Check transceiver to make sure it is securely attached to the network.
LAN system-software fault; aborted, connection resuming	Internal system error. If problem persists, contact Technical Support.
LAN system-software fault; restarted LAN I/F	Internal system error. If problem persists, contact Technical Support.
LAN system-software fault; resuming	Internal system error. If problem persists, contact Technical Support.
LAN transceiver fault; OFF network until fixed	Transceiver or transceiver cable failed or became disconnected. Reattach the cable or replace the transceiver cable. Check SQE test switch if present on transceiver.

Controller Fault	User Action
Local request to send was rejected; discarded request	Internal error. Check that the Ethernet Interface is online. If problem persists, contact Technical Support.
Memory backup fault; may lose configuration/log on restart	Internal error accessing non-volatile device. If problem persists, contact Technical Support. Replace the Ethernet Interface.
Module software corrupted; requesting reload	Catastrophic internal system error. Contact Technical Support.
Module state doesn't permit Comm_Req; discarded	COMM_REQ received when Ethernet Interface cannot process COMM_REQ. Make sure Ethernet Interface is configured and online. Error may occur if the logic application is sending COMM_REQs faster than the Ethernet Interface can process them. Reduce the rate at which COMM_REQs are sent.
Unsupported feature in configuration	An attempt has been made to configure a feature not supported by the Ethernet Interface. Check CPU and Ethernet Interface revisions, order upgrade kit for CPU and/or Ethernet Interface.
Can't locate remote node; discarded request	A specified remote device does not exist on the network. Check that the remote device IP address is correct and that the remote device is functioning properly.
Mailbox Queue full – Comm_req aborted	The CPU is attempting to send COMM_REQs faster than the Ethernet Interface can receive them. The controller logic program should retry the COMM_REQ after a short delay. If the condition persists, the logic application should be revised to reduce the rate at which it sends COMM_REQs to the Ethernet Interface.
Non-critical CPU software event	The CPU is attempting to send mail messages faster than they can be retrieved by the Ethernet Interface; the messages are discarded. This can result in subsequent "Backplane communications with controller fault; lost request" faults.

9.8.1.2 Station Manager

You can use the Station Manager to troubleshoot a problem with the Ethernet Interface, the network or with your application.

Only monitor commands are supported in the first release:

EXS, HELP, LOG, LTIME, NODE, PLCREAD, SOSW, STAT, TALLY and TIME

The LOG, TALLY, and STAT Station Manager commands are especially useful.

- The LOG command provides a complete record of exceptions occurring with the network and Interface.
- The TALLY command provides statistics about operation and performance of the network and Interface. The RXi embedded Ethernet interface supports the C, I, J, L, M, P, U, V, and W tasks.
- The STAT command provides the current status of specific components of the Ethernet interface. The RXi embedded Ethernet interface supports the B, C, I, L, M, U, V and W tasks.

Refer to the *TCP/IP Ethernet Communications for PACSystems Station Manager Manual*, GFK-2225, for information on how to access and use the Station Manager software.

9.8.1.3 Ethernet Interface Status Bits

The Ethernet Interface status bits occupy a single block of either reference memory or I/O variables. The access type and location of the Ethernet Interface Status bits is specified during configuration of the Ethernet Interface. The Ethernet interface updates its bits in the CPU once each controller scan.

The first 16 bits of the block are the LAN Interface Status (LIS) bits. The next 64 bits are the Channel Status bits (2 for each channel).

	Status Bits	Brief Description
LAN Interface Status	1	Port full duplex
	2	Port operating at highest supported speed
	3	Reserved
	4	Reserved
	5	Reserved
	6	Reserved
	7-8	Reserved
	9	Any Channel Error (error on any channel)
	10–12	Reserved
	13	LAN OK
	14	Resource problem
	15	Reserved
	16	LAN Interface OK
Channel Status	17	Channel Open - Channel 1
	18	Reserved – Channel 1
	...	...
	47	Channel Open - Channel 16
	48	Reserved – Channel 16
	49–80	Reserved

9.8.1.4 LAN Interface Status (LIS) Bits

The LAN Interface Status bits (bits 1 – 16) monitor the health of the Ethernet Interface.

Note: Unless the “LAN Interface OK” bit is set (Status Bit 16), the other status bits are invalid.

Bit 1, Port set to Full Duplex: This bit is set to 1 when the port is set to full duplex. Full-duplex or half-duplex operation is automatically negotiated between the Ethernet Interface and its immediately-connected network device, usually a network hub or switch. If this bit is 0, the port is in half-duplex Ethernet mode. This bit is only valid if bit 13 (LAN OK) is 1.

Bit 2, Port Operating at Highest Supported Speed: This bit is set to 1 when the port is operating at its highest supported speed.

Bit 9, Any Channel in Error: This bit (normally 0) indicates one or more of the channels are in error.

Bit 13, LAN OK: This bit is 1 as long as the Ethernet Interface sis able to communicate on the network. If the network becomes inaccessible due to local or network problems, this bit is set to 0. If LAN communication becomes possible again, it is set to 1.

Bit 14, Resource Problem: This bit is set to 1 if the Ethernet Interface software has a resource problem (i.e., lack of data memory). The bit is reset to 0 on a subsequent controller sweep. The Ethernet Interface may or may not be able to continue functioning, depending on the severity of the problem. Look in the Controller Fault Table for details. In addition, the Station Manager STAT B and LOG commands can be used. See the *Station Manager Manual*, GFK-2225, for more information.

Bit 16, LAN Interface OK Bit: This bit is set to 1 by the Ethernet Interface each controller scan. If the Ethernet Interface cannot access the controller, the CPU sets this bit to 0. *When this bit is 0, all other Ethernet Interface Status bits are invalid.*

9.8.1.5 Channel Status Bits

The Channel Status bits provide runtime status information for each communication channel. Each channel has two status bits; the meaning of the channel status bits depends upon the type of communication performed on that channel.

Each Modbus channel has a dedicated status bit:

(Status Bits 17, 19, 21 ... 47) Connection Open Bit: This bit is 1 when a TCP connection exists for the associated channel. The bit is 0 when the connection does not exist or is unused (either never created or has disconnected). The bit is also set to zero when the controller goes to STOP, because all connections are automatically closed upon STOP transition.

(Status Bits 18, 20, 22 ...46, 48–80) Reserved: When a Channel is in use as a Modbus TCP Channel, these bits are not used.

9.8.2 Troubleshooting Common Ethernet Difficulties

Some common Ethernet errors are described below. Ethernet errors are generally indicated in the Controller Fault Table and the Ethernet exception log. As previously explained, Controller Faults generated by the Ethernet interface contain Ethernet exception events within the extra fault data. See the *TCP/IP Communications for PACSystems Station Manager Manual*, GFK-2225 for detailed descriptions of Ethernet exception events.

9.8.2.1 COMM_REQ Fault Errors

When the controller CPU attempts to initiate COMM_REQs to the Ethernet Interface more rapidly than the Ethernet Interface can accept them, the COMM_REQ delivery will fail. The fault output of the COMM_REQ function block will be set and the COMM_REQ will not be delivered to the Ethernet Interface. In this case, the logic program should attempt to initiate the COMM_REQ on another sweep after a very short delay. This condition may arise when the logic Program attempts to initiate greater than 16 COMM_REQs in the same logic sweep.

Sustained heavy COMM_REQ delivery from the CPU to the Ethernet Interface can use a considerable portion of the Ethernet Interface's processing capability. Under heavy COMM_REQ load, the Ethernet Interface may discard some received COMM_REQs until it is once again able to process further COMM_REQs. In such cases, the Ethernet Interface increments the "CmrqDscd" tally; this tally is available via the TALLY C Station Manager command.

Under sustained extremely heavy COMM_REQ load, the Ethernet Interface may not respond to Station Manager commands and possibly some network communications. A COMM_REQ fault may be logged in the Controller Fault Table (see Controller Fault Table Descriptions, earlier in this chapter.) If this occurs, first switch the controller CPU to STOP mode, which ceases COMM_REQ delivery in order to resume normal Ethernet operation. Then modify the logic application to reduce the COMM_REQ traffic to a manageable level.

9.8.2.2 Controller Timeout Errors

Controller timeout errors may occur when the SRTP traffic to the Ethernet Interface exceeds the controller's ability to process the requests, or when the controller is unable to deliver mail to the Ethernet Interface. Controller Timeout errors will take down an SRTP Server connection; in this case, the remote SRTP client must establish a new SRTP connection to the Ethernet Interface.

This error is indicated in the Controller Fault Table as:

"Backplane communication with controller fault; lost request"
with exception Event = 8, Entry 2 = 8

These errors may also be accompanied by any of the following:

"Backplane communication with controller fault; lost request"
with exception Event = 8, Entry 2 = 6; location = Ethernet Interface

"LAN system-software fault; resuming"
with exception Event = 8, Entry 2 = 16; location = Ethernet Interface

"Non-critical CPU software event"
status code (bytes 5-8) = 80 3a 00 12; location = CPU module

The controller Timeout condition occurs when the CPU cannot process requests within a specified timeout period. The remedy is to reduce the rate of requests, or increase the processing capacity in the controller.

Cause	*Corrective Action*
Heavy COMM_REQ traffic.	Reduce the rate at which the logic application sends COMM_REQs to the Ethernet Interface.
Heavy SRTP traffic.	Reduce the size, number, or frequency of SRTP requests at the remote SRTP client.
Long controller sweep time.	Modify the controller application to reduce the controller sweep time.
Controller Communication Window set to LIMITED mode.	Change to RUN-TO-COMPLETION mode.

9.8.2.3 Station Manager Lockout under Heavy Load

Sustained heavy SRTP Server load can utilize all processing resources within the Ethernet interface, effectively locking out the Station Manager function. The Station Manager appears inoperative under either local or remote operation. The Ethernet interface always gives higher priority to data communication functions than to the Station Manager. When the processing load is reduced, the Station Manager becomes operative once again.

This condition is not reported to the Controller Fault Table or Ethernet exception log.

9.8.2.4 Ping Restrictions

To conserve network data buffer resources, the CPU process only one ICMP control message at a time. An ICMP Echo (ping) request that arrives while the CPU is processing another ICMP control message is discarded. When multiple remote hosts attempt to ping the CPU at the same time, some individual ping requests may be ignored depending upon the timing of the ping requests on the network.

Discarded ping requests are not reported to the Controller Fault table or Ethernet exception log.

9.8.2.5 SRTP and Modbus TCP Connection Timeout

When the Ethernet Interface is abruptly disconnected from a remote SRTP or Modbus TCP device (for example, by disconnecting the Ethernet cable), the underlying TCP connection attempts to re-establish communication. By default, the underlying TCP connection in the Ethernet Interface remains open for 7 minutes while TCP attempts to reconnect. During this interval, the SRTP or Modbus TCP connection is unavailable. If all the SRTP or Modbus TCP connections in the Ethernet Interface are in use or otherwise unavailable, a new SRTP or Modbus TCP server connection must wait until an existing SRTP or Modbus TCP connection times out. If the SRTP server connection was used by the Programmer, any new Programmer connection is restricted to Monitor operation until the previous connection times out and is cleaned up.

For details, see 9.4.1, SRTP Inactivity Timeout."

Note that the TCP connection timeout interval applies to all TCP-based connections at this Ethernet interface. This includes all SRTP, Modbus TCP, FTP, and web server communications.

The underlying TCP connection timeout is normal expected behavior, and is consistent with our other controller products.

9.8.2.6 Sluggish Programmer Response after Network Disruption

The network programmer attempts to use a special "privileged" SRTP server connection at the Ethernet Interface in order to establish and maintain connection even under heavy load due SRTP connections. The Ethernet Interface allows three privileged connections. When the maximum number of privileged connections is in use, no other privileged connections are permitted until a current privileged connection is terminated. This normally occurs when the network programmer disconnects from the target controller.

As described above under "SRTP Connection Timeout", when the programmer-controller network connection is abruptly broken (not the orderly termination performed during disconnection), the SRTP server connection and its underlying TCP connection remain alive until either an SRTP inactivity timeout occurs (20 –30 seconds), or the TCP connection times out (about 7 minutes). If the maximum privileged connections are in use and the programmer reconnects during this interval, it obtains a new, non-privileged connection. Under heavy load at the Ethernet Interface, the programmer may experience sluggish response over this non-privileged connection. If this occurs, you can manually disconnect and reconnect the programmer after the previous connection has timed out. Upon reconnection, the programmer should once again obtain a privileged connection.

9.8.3 COMM_REQ Flooding Can Interrupt Normal Operation

The controller logic application program should generally wait for a response from each COMM_REQ function block before activating another COMM_REQ function block to the same endpoint. Extremely heavy COMM_REQ delivery loading, such as activating the same COMM_REQ every logic sweep, can prevent normal SRTP, Modbus, and Station Manager operation. During such loading, the Ethernet LAN LED may be frozen. Under extreme COMM_REQ loading, the Ethernet interface may automatically restart.

9.8.4 Channels Operation Depends Upon Controller Input Scanning

Communication channels operation always includes updating the Channel Status Bits (located within the Ethernet Status data) into controller memory, which occurs when the controller scans inputs from the Ethernet module. At least one controller input scan must occur for each data transfer on a channel, so the channel can run no faster than the controller scans the Ethernet Status data. When the Ethernet interface is configured to use an I/O Scan Set than runs more slowly than the controller sweep, each channel must wait until the next time that its scan set runs to transfer its Channel Status bits. This can reduce channels performance.

If the Ethernet interface is configured to use an inactive I/O Scan Set, the Channels Status bits will not be transferred and channel operations will not complete.

Chapter 10. PROFINET Controller Overview

The embedded PNC provides an integrated redundant PROFINET I/O interface. It provides all the functions, services, and protocols required for certification as a PROFINET IO Version 2.2 IO Controller, running at both 100Mbps and 1Gbps.

The two PNC ports support 10/100/1000Mbps copper connections. The network can include media interfaces of more than one type if using an external programmable switch. PROFINET communications on the network require 100 or 1000 Mbps link speed. The 10Mbps speed cannot be used for PROFINET communications. However, 10Mbps can be used for other types of Ethernet traffic, such as ping.

Features of the RXi PNC include:

- Full programming and configuration services for the PNC, GE Intelligent Platforms' PROFINET Scanners (PNSs) and third-party IO-Devices using PROFICY Machine Edition software.
- Support for star, ring, and daisy-chain/line network topologies.
- No external switches are required for any ring and line topologies.
- Support for media redundancy
- Provides an internal clock for time-stamped diagnostics entries.

10.1 Ethernet Network Ports

The PNC connects to a PROFINET network via one or both of its two external switch ports. For details, see "2.4.3, Connecting to a PROFINET Network."

The PROFINET protocol supported by the PACSystems RXi can be sent and received over either of the two external ports.

The Controller is assigned seven Ethernet MAC addresses: one for each of the three external Ethernet ports and four for the internal switch.

Each external switch port has an associated link-up/link-down status bit that can be monitored to check the operating status of the port (see 11.8.2 "PROFINET Controller Status Reporting" for information about the PNC status bits.

10.2 PROFINET Networks for PACSystems

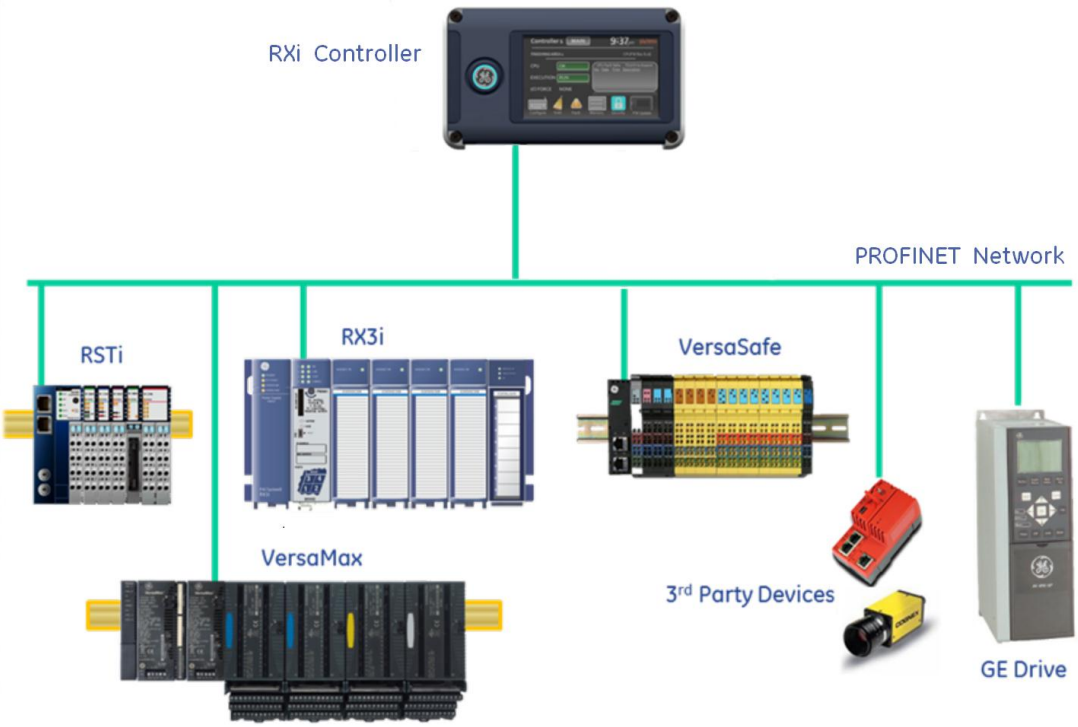
PROFINET is an open standard for industrial automation that is based on Industrial Ethernet. The PROFINET IO framework allows the creation of I/O data exchanges between controllers and distributed devices. It also allows configuration, parameterization, and diagnostics communication between controllers and devices.

Note: The PNC operates only in autonegotiate mode. All PROFINET bus devices and switches that are connected to the PNC should be configured to use autonegotiation.

10.2.1 Basic System: One RXi Controller using a single port

Components of the RXi PROFINET network consist of a PACSystems RXi Controller communicating with IO Devices on the PROFINET bus. IO Devices on the network can include GE Intelligent Platforms PNS modules and a wide range of third-party devices. The example below shows a basic system with one RXi Controller, and a PROFINET network with GE Intelligent Platforms IO-Scanners and third-party IO-Devices.

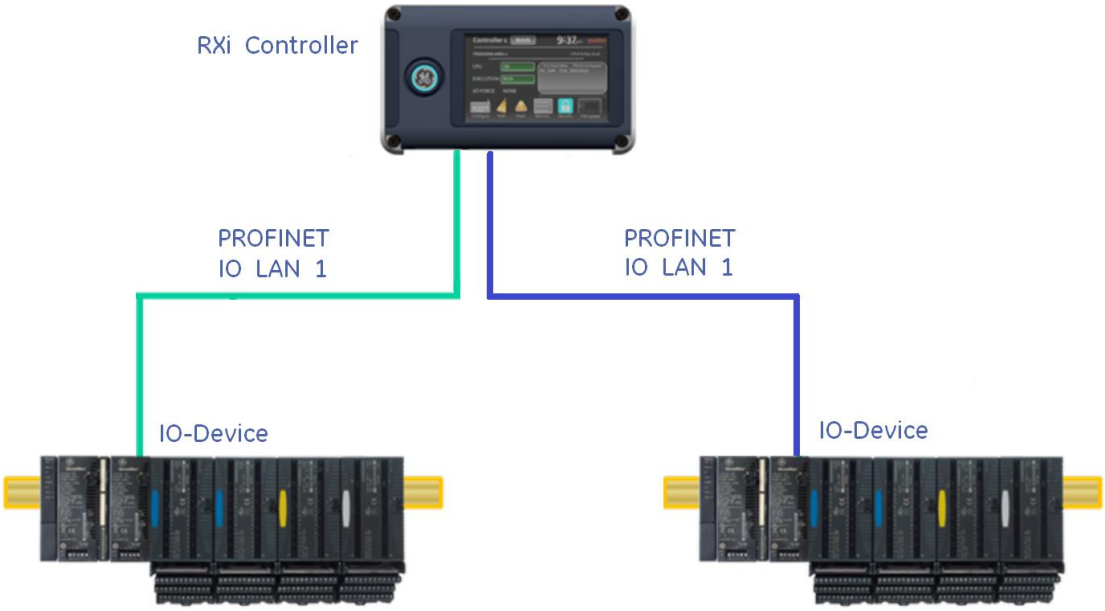
An RXi Controller can control up to 128 devices. The PROFINET Scanners can interface multiple devices such as discrete and analog modules to the PROFINET network.



10.2.2 Basic System: One RXi Controller using Multiple Ports

The illustration below shows a basic system consisting of an RXi Controller node controlling one PROFINET network. The network can connect up to 128 compatible IO-Devices, including any combination of GE Intelligent Platforms PROFINET Scanners and third-party IO-Devices.

This example shows an RXi Controller that is directly connected to two separate IO-Devices in a star topology. Although each IO-Device is connected to a separate Ethernet port on the PNC, they are all on the same network. The IO-Devices in this example are VersaMax PROFINET Scanners, but other types of IO-Devices can also be used.

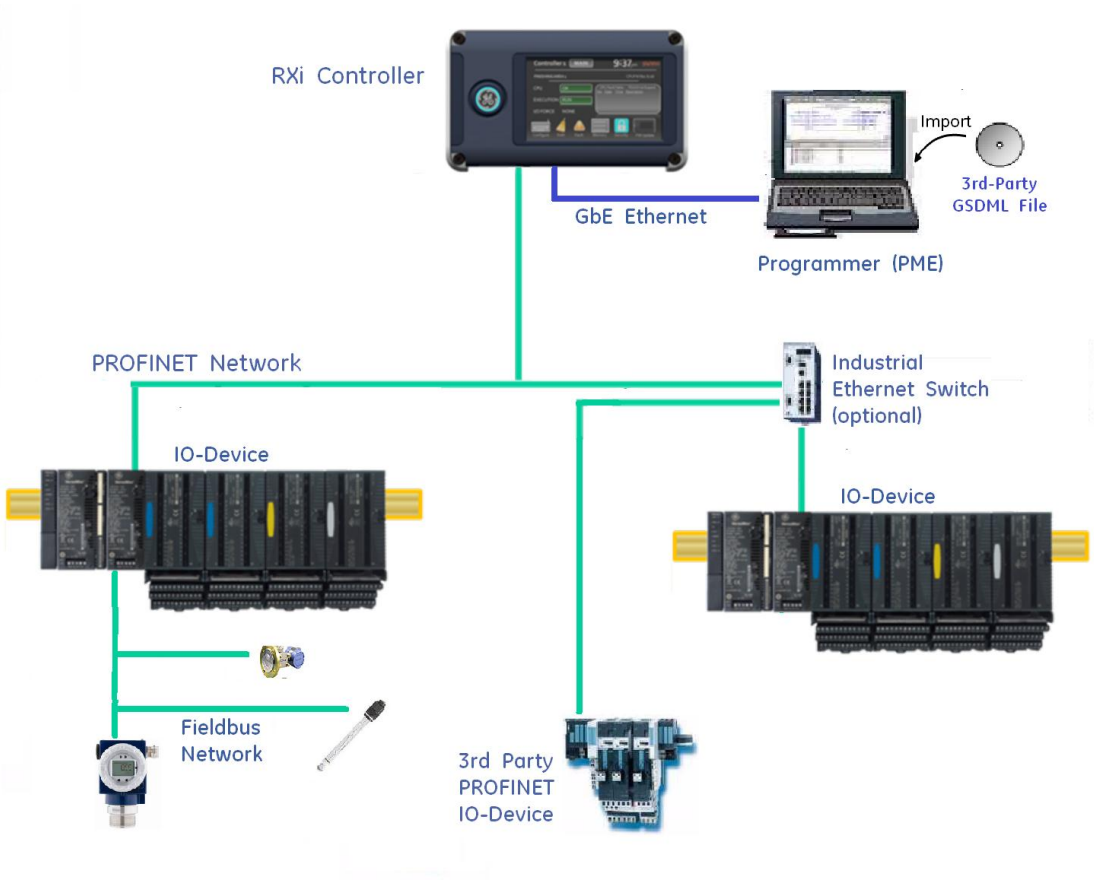


10.2.3 Basic System: Third-Party Devices and PME Programmer

Third-party IO Devices can be used with the PNC if their manufacturer provides a GSDML file that can be imported into Proficy Machine Edition. The GSDML file defines the characteristics of the IO-Device and its I/O modules. Importing a third-party IO-Device GSDML file and configuring third-party IO-Devices are described in 4.2.6, “Adding a Third-Party IO-Device to a LAN” and in the PME online help.

After receiving a third-party device’s configuration, the PNC connects to the third-party IO-Device if the device is available, transfers the configuration to the device, and starts exchanging I/O and alarm data with the device.

The following illustration shows a programmer connection (for configuration, user logic programming, and monitoring), the concept of GSDML import, an optional external Ethernet switch, and the ability to connect field buses to a PROFINET IO-Device. Third-party IO-Devices that have only one Ethernet port may require the use of an external switch.



10.3 Glossary of PROFINET Terms

AR	**A**pplication **R**elationship. PROFINET term for a relationship that is established between an IO-Controller/Supervisor and IO-Device. For any data to be exchanged between an IO-Controller/Supervisor and a given IO-Device, an Application Relationship must be established. Within the Application Relationship, various Communication Relationships are then established for the different types of data to be exchanged.
Broadcast	In Ethernet, the transmission of a network message to all hosts on the network.
CR	**C**ommunication **R**elationship. PROFINET term for a channel that is established within an Application Relationship to transfer specific data between an IO-Controller/Supervisor and a given IO-Device. Multiple CRs are established within an AR to transfer data.
DAP	**D**evice **A**ccess **P**oint. This access point is used to address an IO-Device as an entity.
Gratuitous ARPs	An Address Resolution Protocol (ARP) request sent by the host to resolve its own IP address.
GSDML	**G**eneral **S**tation **D**escription **M**arkup **L**anguage - definition of PROFINET Device Characteristics.
IOC	PROFINET **IO**-**C**ontroller
IOD	PROFINET **IO**-**D**evice
IOCR	**I**nput **O**utput **C**ommunication **R**elationship – describes the type (input/output) and amount of I/O data to be transferred, the sequence of the transfers and the transfer cycle between a PROFINET IO-Controller (or IO-Supervisor) and a PROFINET IO-Device.
IOCS	PROFINET **I**nput/**O**utput **C**onsumer **S**tatus is transmitted on the PROFINET network to provide feedback on Input Data for an IO controller and Output Data for an IO device.
IOPS	PROFINET **I**nput/**O**utput **P**rovider **S**tatus is transmitted on the PROFINET network to provide feedback on Output Data for an IO controller and the Input Data for an IO device.
IOxS	PROFINET abbreviation for the IOCS and/or IOPS (see above).
LLDP	**L**ink **L**ayer **D**iscovery **P**rotocol. IEEE standardized protocol used by network devices to advertise their identity and capabilities.
MRC	**M**edia **R**edundancy **C**lient. Within Media Redundancy Protocol, an MRC is responsible for helping the MRM detect breaks/no breaks in the ring.
MRM	**M**edia **R**edundancy **M**anager. Within Media Redundancy Protocol, an MRM is responsible for ensuring that the ring does not have a closed loop, while simultaneously ensuring maximal connectivity between nodes on the ring.
MRP	**M**edia **R**edundancy **P**rotocol. An Ethernet protocol that provides redundant paths for PROFINET-IO cyclic traffic by supporting a ring topology.
Multicast	In Ethernet, the transmission of a network message to all hosts within a host group.
NMS	**N**etwork **M**anagement **S**ystem. Executes applications that monitor and control managed devices in an SNMP-managed network.
Phase	If the IOCR Update Period is greater than the Send Clock time, the Update Period is divided into multiple phases where each phase is equal to one Send Clock.
CPU Node	In a PACSystems RXi PROFINET network, a CPU Node is a node that has a PACSystems RXi Controller that is connected to the PROFINET network.
RDO	**R**ecord **D**ata **O**bject. Services used to read and write structured data stored in a PROFINET IO-Device.

Reduction Ratio	Along with Send Clock determines the Update Period for a PROFINET cyclic data transfer between two devices (see IOCR). The Update Period equals the Reduction Ratio multiplied by the Send Clock time. For example, if the Reduction Ratio is 4 and the Send Clock is 1ms, the Update Period is 4ms.
Remote Node	For an RXi PROFINET network, a Remote Node is any PROFINET IO-Device, such as a rack of I/O modules with a Remote Scanner or a third party PROFINET IO-Device.
RTA	**R**eal-**T**ime **A**cyclic. A PROFINET-IO Mechanism used to exchange non-periodic data such as alarms.
RTC	**R**eal-**T**ime **C**yclic. A PROFINET-IO Mechanism used to exchange input and output data.
Send Clock	Value between 1 and 128 inclusive in 31.25 µs units (equivalent to a range of 31.25 µs to 4 ms) used to calculate the Update Period for a PROFINET cyclic data transfer between two devices (see IOCR). The Send Clock is the basis for all other scheduling parameters.
Send Offset	The time to delay a scheduled PROFINET cyclic data transfer frame. Measured in nanoseconds from 0 to 3,999,999. Must be less than the Send Clock time.
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol. UDP-based network protocol that facilitates the exchange of management information between network devices.
Submodule	PROFINET-IO representation of the smallest configurable entity of a PROFINET Module.
Unicast	In Ethernet, the transmission of a network message to an individual host.
Update Period	The time between PROFINET cyclic data transfers between an IO-Controller and an IO-Device.

Chapter 11.PROFINET Controller Operation

This chapter describes:

- PROFINET Operation Overview
  - PROFINET Communications
  - Application Relationships
  - Types of PROFINET Communications
  - External Switch VLAN Priority Settings
- Operations of the PNC
  - Duplicate Device IP Address Detection
  - Duplicate Controller IP Address Detection
  - Resolving Duplicate IP Addresses
- I/O Scan Timing
- RXi CPU Operations for PROFINET
  - Reference ID Variables for the RXi Application
  - The PNIO_DEV_COMM function block
  - DO I/O for Remote I/O Modules
  - Scan Set I/O for Remote I/O Modules
  - RXi CPU Defaults Inputs
  - RXi CPU Defaults Outputs

11.1 PROFINET Operation Overview

An RXi Controller uses PROFINET communications for data exchange. The same network can also be used for basic Ethernet communications, but use of a separate Ethernet LAN and RXi Ethernet interface is recommended for most applications.

A PROFINET network can include three types of devices:

PROFINET IO-Controller	The RXi Controller operates as an IO-Controller. It is a controlling device that is associated with one or more IO-Devices.
PROFINET IO-Device	A PROFINET IO-Device is a distributed I/O Device that is coupled to a PROFINET IO-Controller via PROFINET.
PROFINET IO-Supervisor	An IO-Supervisor can be a programming device, a computer, or an HMI device. The PROFINET IO-Supervisor is typically used for commissioning or diagnostics.

11.1.1 PROFINET Communications

Communications on an RXi PROFINET network use the standard PROFINET communications described in this section.

11.1.2 Application Relationships

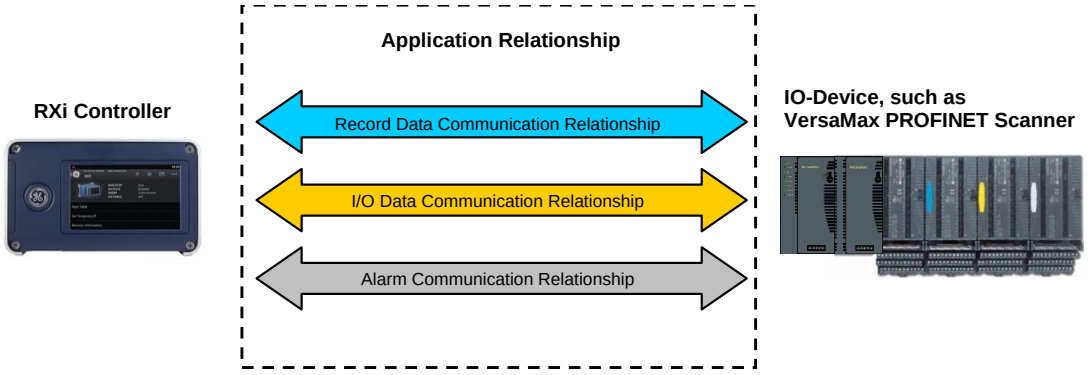
Before an RXi PROFINET IO-Controller can exchange data with a PROFINET IO-Device, an Application Relationship (connection) must be established between the devices. The RXi PROFINET IO-Controller automatically sets up the correct number and types of Application Relationship and Communication Relationship channels (see below) based on its Proficy Machine Edition configuration. Usually, only one Application Relationship is established per IO-Device.

Communication Relationships within an Application Relationship

Within each Application Relationship, the RXi PROFINET IO-Controller establishes the following types of Communication Relationships (CRs):

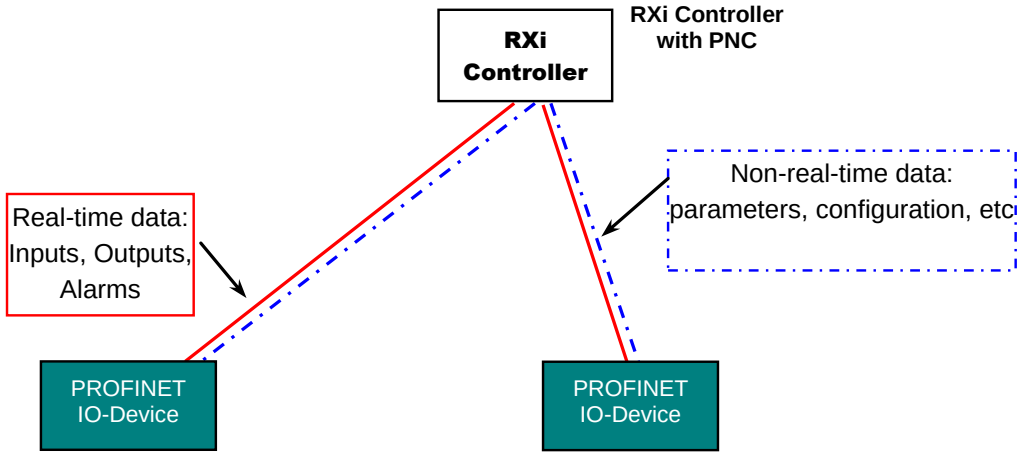
- **Record Data CRs** – always the first to be established within an Application Relationship. Record Data Communication Relationships are used for non-real-time transfers of data records such as startup parameter data, diagnostics data, identification data, and configuration data.
- **IO CRs** – used for the real-time, cyclic transfer of I/O data
- **Alarm CR** – used for real-time, acyclic transfer of alarms and events

The illustration below represents an Application Relationship between a PACSystems Controller and an IO-Device. In this example, the IO-Device is a VersaMax PNS with VersaMax I/O modules, but the same principles apply for all IO-Controllers and IO-Devices.



11.1.3 Types of PROFINET Communications

The PNC uses two types of PROFINET communication transfers: real-time and non-real-time. The illustration below shows real-time communications as solid lines and non-real-time communications as dashed lines.



- **Real-Time (RT) communication**: PROFINET real-time communication is used for time-sensitive data. A PROFINET IO-Controller and PROFINET IO-Device use two types of real-time communications to exchange data: cyclic communication and acyclic communication:
  - Real-time Cyclic communication is used to periodically transfer the application's input and output data. Cyclic communication occurs each PROFINET IO production cycle.
  - Real-time Acyclic communication is used to transfer non-periodic data such as alarms. Acyclic communication occurs only when needed.
- **Non-Real-Time (NRT) communication**: PROFINET non-real-time communication is used for less time-sensitive data such as configuration, parameterization, diagnostics, and identification data.

11.1.4 External Switch VLAN Priority Settings

The PROFINET-IO specification indicates the VLAN priorities for each type of Ethernet traffic that originates from a PROFINET node. VLAN priorities range from 0 to 7, with 7 being the highest.

The switch on the PNC supports just four traffic classes, giving four levels of preference. Incoming traffic without a VLAN priority is assigned to the lowest priority traffic class. The table below lists the VLAN priorities, and their corresponding priorities in the PNC:

VLAN Priority	PNC Priority	Ethernet Traffic	Description
7	Highest priority	MRP	Media Redundancy
6	Second-highest priority	RT_CLASS_1	Cyclic PROFINET IO
		High Priority RTA_CLASS_1	High-Priority PROFINET Alarms
5	Third-highest priority	Low Priority RTA_CLASS_1	Low-Priority PROFINET Alarms
4, 3, 2, 1	Lowest priority	(reserved)	(reserved)
0		IP DCP	Device Discovery and Configuration

If a system includes external switches, these switches must be configured to match the VLAN Priority groupings listed above for the PNC.

11.2 Operations of the PROFINET Controller in the RXi System

The PNC performs the following operations:

- Consumes PROFINET IO-Device configuration from the CPU and transfers it to the IO-Devices over the PROFINET network.
- Consumes input data from each PROFINET IO-Device and makes that data available to the CPU during the CPU's input scan.
- Produces the output data that it receives from the CPU during the CPU's output scan to each PROFINET IO-Device.
- Receives PROFINET alarms and diagnostics from PROFINET IO-Devices and converts them to a PACSystems format.
- Automatically converts between the little-endian data format recognized by the RXi CPU and the big-endian format used for PROFINET communications.
- Checks for duplicate IP addresses as described below.

11.2.1 Duplicate PROFINET Device IP Address

The PNC will detect an IP address conflict between a device that it is configured to communicate with and another device in two situations:

- First, duplicates are detected when the PNC is trying to initially establish communications with the configured PROFINET IO Device. During the connection sequence, the PNC queries the network to see whether any other node has the same IP address as the configured device.
- Second, duplicates are detected when a network device announces its presence[9] on the network and that device's IP address is identical to that of a PROFINET IO Device that the PNC is currently communicating with.

In both cases, the PNC attempts to establish or maintain the connection and logs a Duplicate IP Address Detected fault for the device. The PNC then periodically queries the network for resolution of the IP address conflict. If the IP address conflict is resolved, the PNC logs a Duplicate IP Address Resolved fault for the device.

[9] The PROFINET Controller uses the ARP protocol to detect duplicate IP addresses. Devices that issue a gratuitous ARP to announce their presence on the network are detected.

11.2.2 Duplicate PROFINET Controller IP Address

The PNC detects that a network device has the same IP address as its own during powerup, when a new hardware configuration is downloaded from the programmer, and during operation when a device with a conflicting IP address announces its presence[10] on the network.

When a duplicate is detected during powerup, PNC reset, or new hardware configuration store, the PNC:

- logs a Duplicate IP Address Detected fault for itself,
- does not connect to any configured PROFINET IO Devices,
- periodically queries the network for resolution of the IP address conflict.

When a duplicate IP address is detected after the PNC has established connection to configured IO Devices, the PNC:

- logs a Duplicate IP Address Detected fault for itself,
- disconnects from all currently-connected PROFINET IO Devices,
- logs a Loss of Device fault for each PROFINET IO Device that was disconnected,

When the IP conflict is resolved, the PNC:

- logs a Duplicate IP Address Resolved fault for itself,
- attempts to re-connect all configured PROFINET IO Devices,
- logs an Addition of Device fault for each connected PROFINET IO Device to indicate that device is back online.

Note: Power cycling an RXi Controller with the same IP address as another node on the network will result in two Duplicate IP Address Detected faults in the I/O Fault table. This is normal behavior that occurs because the PNC retains IP parameters through a power cycle and attempts to exist on the network before receiving a new configuration from the CPU. The first fault occurs before the PNC receives the new configuration and the second fault occurs after the PNC receives its new configuration. Both faults result in the PNC not attempting to connect to the network.

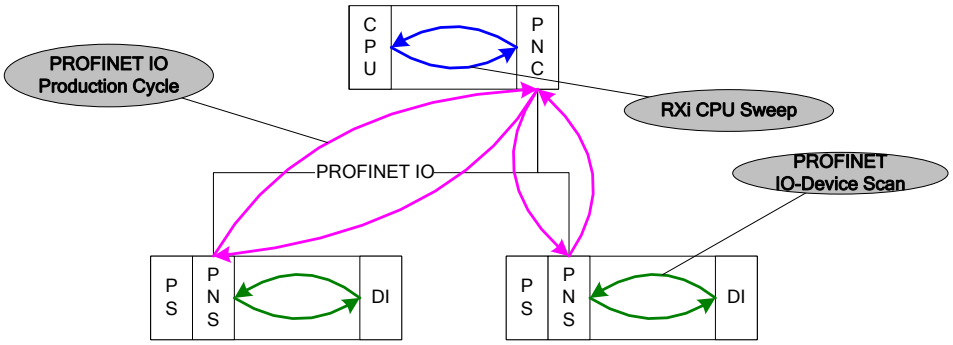
Resolving Duplicate IP Addresses

When an IP address conflict exists, IP-based network communication with the device(s) may be disrupted. The IP address conflict should be resolved by disconnecting one of the offending devices from the network or assigning each a unique address. The Duplicate IP Address Detected fault lists the MAC address of the offending devices in bytes 8 – 13 and 14 – 19 of the Fault Extra Data. The Discovery and Configuration Protocol (DCP) tool in Proficy Machine Edition may be useful to identify PROFINET devices on the network with conflicting IP addresses.

11.3 I/O Scanning

In the PACSystems RXi PROFINET network, multiple I/O cycles run asynchronously and independently. The example below illustrates typical cycles in a system with an RXi Controller and VersaMax PNS modules used as IO-Devices. Cycles may be different for other GE Intelligent Platforms scanners and third-party devices.

[10] The PROFINET Controller uses the ARP protocol to detect duplicate IP addresses. Devices that issue a gratuitous ARP to announce their presence on the network are detected.



- **PROFINET IO-Device Scan**: in this example, each VersaMax PNS scans all the modules in its node as quickly as possible. The PNS stores the modules' input data into its internal memory. On each PNS output scan, the Scanner writes the output data from its internal memory to the modules in its node.
- **PROFINET IO Production Cycle**: each PNC and each IO-Device publishes data from its internal memory onto the network at each scheduled PROFINET production cycle (note: production cycles between IO-Controllers and IO-Devices are not synchronized, each publishes at its configured update rate independently). The PNC publishes output data received from the RXi CPU to each IO-Device, and the IO-Device publishes input data from its memory to the PNC.
- **RXi CPU Sweep**: the sweep includes both an input scan and an output scan. The CPU input scan retrieves the current input data from the PNC. This input data is then available for use by the application logic. After the logic solution, the CPU output scan writes the outputs to the PNC.
- **3rd party devices:** The conveyance of I/O data between an I/O module and the PROFINET IO network is device dependent. Third party manufacturer documentation should be referred to for specifics for a particular device.

11.4 Data Coherency

In a PACSystems RXi PROFINET network, it is important to note that I/O data coherency is at the PROFINET submodule level. The PNC coherently transfers I/O data to and from the CPU on a PROFINET submodule basis. This means that output data from a single CPU output scan for multiple PROFINET submodules may not be transferred during the same PROFINET IO production cycle. Conversely, input data consumed from a single PROFINET IO cycle by the PNC may not be consumed during a single CPU input scan.

11.5 Performance Factors

There are many factors that affect the timing of I/O as it flows through the system. Primary factors include:

- CPU Sweep Time
- Configured PROFINET IO Update Rate(s)
- Number of PROFINET IO Devices
- Number of IO modules
- Network latency and loading (for example, switching hardware, additional non-PROFINET network traffic)
- I/O Module Filter Times

When designing a PROFINET IO system, consider and weigh these factors appropriately to achieve an optimal IO system for the application.

11.6 PROFINET IO Update Rate Configuration

Selecting PROFINET IO update rates is one of the primary means for adjusting performance of the system. Consider the following when choosing an appropriate value.

- In general, for most applications, there is little benefit to configuring PROFINET IO update rates faster than half the CPU Sweep time. Scheduling PROFINET IO update rates faster than required by the application creates unnecessary loading on the network, PNC, and PROFINET IO devices.
- Keep in mind that transferring IO over the PROFINET IO network could take up to one complete PROFINET update cycle time for the transfer to actually occur. Since PROFINET IO production is asynchronous to the source of the produced data, in the worst case the new production data could miss a PROFINET IO production cycle, and thus must await the next cycle. Note, this means for an I/O loopback situation where an application asserts an output and expects to see the output echoed on another input, there are two PROFINET transfers involved, therefore it may take two PROFINET IO production cycles (one for each data transfer).
- It is possible that the RXi CPU, through application logic actions, can update output data for a Remote IO Module faster than the usual I/O update rate of that remote I/O module. The application logic must be careful not to update output data faster than the scanning of the Remote IO Module (which is a function of both the PROFINET IO Update Rate and IO-Device Scan). Otherwise, output data from the CPU may not transfer to the Remote IO Module before being overwritten by new output data from the application logic.

11.7 RXi CPU Operations for PROFINET

This section describes several CPU functions as related to their operation when used with a PROFINET network.

In addition, the RXi CPU and Proficy Machine Edition provide special tools for use in systems with PROFINET:

- Reference ID Variables (RIVs)
- PNIO_DEV_COMM function block

11.7.1 Reference ID Variables for the RXi Application

RIVs are available to the application logic to provide a simple symbolic reference to an entity. The following RIV types are defined for use with PNCs and IO-Devices.

RIVs are assigned in Proficy Machine Edition by editing the Properties for a PNC or IO-Device in the hardware configuration.

Reference ID Variable Data Types

Data Type	*Associated With*
PNIO_CONTROLLER_REF	PROFINET Controller
PNIO_DEVICE_REF	PROFINET IO-Device

PNIO_CONTROLLER_REF Variable

The PNC in a hardware configuration can have a PNIO_CONTROLLER_REF variable assigned to it.

When assigned, it is linked to the PNC and its value cannot be changed. If a linked PNIO_CONTROLLER_REF variable is present, the application logic and hardware configuration are coupled. The name of the PNIO_CONTROLLER_REF linked variable corresponds to the controller's device name used in the hardware configuration to identify the module on the PROFINET network. If the PNIO_CONTROLLER_REF variable is renamed, Proficy Machine Edition will make sure all uses of that variable in logic indicate the new variable name.

Unlinked PNIO_CONTROLLER_REF variables can be passed to the IN and Q parameters of the MOVE_DATA function block. Linked PNIO_CONTROLLER_REF variables can only be passed to the IN parameter of the MOVE_DATA function block.

PNIO_DEVICE_REF Variable

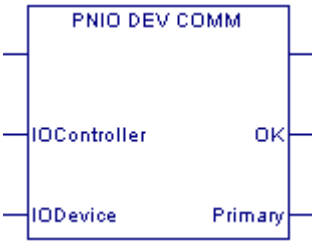
An RIV of type PNIO_DEVICE_REF uniquely identifies a PROFINET IO-Device. It is an unsigned integer in the range of 1 – 255.

Each PROFINET IO-Device in an RXi hardware configuration can have a PNIO_DEVICE_REF variable assigned to it. When assigned, it is linked to a PROFINET IO-Device. When a linked PNIO_DEVICE_REF is present, the logic and hardware configuration are coupled. The name of the PNIO_DEVICE_REF linked variable corresponds to a combination of the LAN ID and the device name used to identify the IO-Device on that LAN. If the PNIO_DEVICE_REF variable is renamed, Proficy Machine Edition will make sure all uses of that variable in logic indicate the new variable name.

Unlinked PNIO_DEVICE_REF variables can be passed to the IN and Q parameters of the MOVE_DATA function block. Linked PNIO_DEVICE_REF variables can only be passed to the IN parameter of the MOVE_DATA function block.

11.7.2 PNIO_DEV_COMM Function Block

The PNIO_DEV_COMM function block monitors communications between a specified PNC and a specified IO-Device.



PNIO_DEV_COMM can be used by the application logic to take a corrective action or turn on an indicator if a specific device fails. It might also be used by a custom HMI to show which PROFINET IO-Device connections are currently established.

It is recommended that the *All Devices Connected* status bit be checked first to determine whether all devices belonging to the PNC are functioning. If this bit is 0, indicating that one or more devices is not OK, the PNIO_DEV_COMM function block can then be used to determine which specific devices are not communicating. For details on this status bit, refer to "11.8.2, PROFINET Controller Status Reporting."

Parameters and Outputs of PNIO_DEV_COMM

PNIO_DEV_COMM returns a Boolean indication of whether or not a given PNC is currently communicating with a specified IO-Device. The PNC is identified by the IO Controller input parameter, which is a PNIO_CONTROLLER_REF data type. The IO-Device is identified by the IODevice input parameter, which is a PNIO_DEVICE_REF data type.

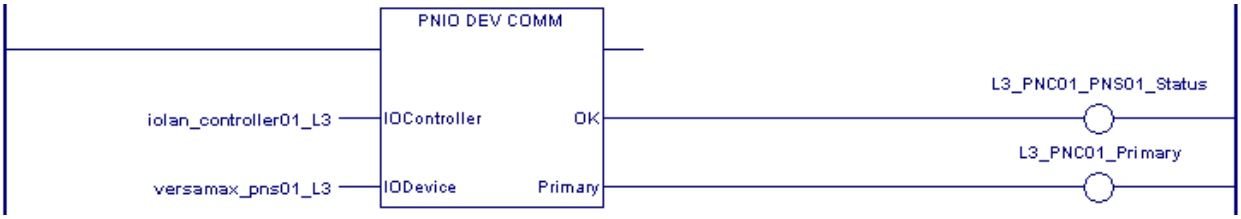
PNIO_DEV_COMM has two Boolean outputs (in addition to ENO) labeled OK and Primary. OK is set ON/true if the PNC is successfully communicating with the IO-Device, otherwise it is OFF/false.

The application logic must identify the PNC and the IO-Device in a symbolic manner, passing appropriate Reference ID Variables (see the following example) to the corresponding input parameters.

Example

In the following sample logic, the RIV *iolan_controller01_L3* is assigned to the PNC and the RIV *versamax_pns01_L3* is assigned to an IO-Device.

If the *iolan_controller01_L3* PNC is communicating with the *versamax_pns01_L3* IO-Device, the Bool variable *LC_PNC01_PNS01_Status* is set on. In a simplex (non-redundant) system, Primary is set to On if OK is set to on. .



11.7.3 DO I/O for Remote IO Modules

In addition to the normal I/O updates that occur during the RXi CPU's scan, the application program in the CPU can use the DO I/O function to access the I/O data associated with Remote IO Modules on the PROFINET network during the logic portion of the CPU sweep. The DO I/O function can obtain or update the most recent I/O data that is being consumed from or published to the Remote IO Module by the PROFINET IO Controller. The DO I/O function can also be used to obtain data associated with the PNC itself.

It is important to remember that the I/O data being read or written by the DO I/O function is data currently being stored by the PNC in its memory. Executing a DO I/O function from the CPU does not cause additional data to be produced or consumed on the PROFINET NETWORK. Updates of the actual Remote IO Module I/O data occur during the configured PROFINET cyclic scanning schedule. The DO I/O function provides the benefit of immediately updating I/O data at the PNC, as opposed to waiting for the next normal RXi I/O scan. The DO I/O function can also be used to obtain the PNC's latest input status data values (see 11.8 "PROFINET Controller Diagnostics").

11.7.4 Scan Set I/O for Remote I/O Modules

The Scan Set I/O function of the RXi CPU requests the scanning of remote modules that are members of a configured scan set. The Scan Set I/O function operates like a DO I/O function, with the added ability to identify and group the modules to be scanned. Modules are grouped according to their configured scan sets. As with the DO I/O function, the Scan Set I/O function updates/consumes data stored in the local PNC; it does not directly update IO-Devices in remote nodes.

11.7.5 I/O Defaults Operation

11.7.5.1 RXi CPU Defaults - Inputs

The CPU defaults input data from remote I/O modules under the following conditions:

- the remote module signals its input data is no longer valid (e.g. due to local module fault)
- the remote module is removed from the network
- the remote module loses power or fails
- the PROFINET network connection (Application Relationship) associated with the input data is lost.

The CPU defaults the input values of an input module based on the input default state configured for the I/O Device. Inputs may be configured to either Hold Last State or Force Off (zero).

If a PNC loses connections to an I/O device, the following actions occur:

1. The CPU defaults the input values of all input modules assigned to it within the IO-Device.
2. The PNC logs a Loss of Device fault in the CPU's I/O fault table.
3. The I/O point fault contacts for the I/O references and variables associated with the IO-Device are set to the faulted state (if point faults are enabled on the CPU).

11.7.5.2 RXi CPU Defaults - Outputs

Outputs default when the CPU is no longer providing the data (e.g. when CPU has IO disabled). I/O modules control their own output defaults based on either their configuration or on a fixed output default behavior (typically 0) if the module's output default behavior is not configurable.

11.8 PROFINET Controller Diagnostics

This section describes:

- Problems During Powerup
- Status Reporting
- Fault Contacts
- PROFINET I/O Alarms
- PROFINET Controller Faults in the RXi Fault Tables
  - Clearing the RXi Fault Tables
  - Faults Reported to the RXi Controller Fault Table
  - Faults Reported to the RXi I/O Fault Table

11.8.1 Powerup and Reset

During powerup and reset, the PNC runs diagnostics and initializes its hardware components. When the necessary hardware components have been initialized and tested, the PNC transitions to normal operation.

11.8.2 PROFINET Controller Status Reporting

The PNC provides 32 bits of status information to a configured location in the RXi CPU's reference memory.

The status data consists of the Module OK bit, which indicates the health of the PNC itself, a status bit for each external port, and a bit that indicates the connection status of the configured devices.

All Status bits are active high. The status location may be configured in %I, %Q, %AI, %AQ, %R, %G, %T, %M or %W or I/O Variable reference memory in the RXi CPU.

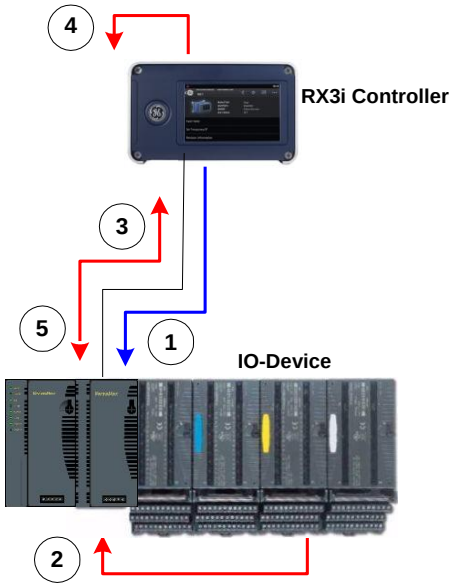
Bit	Name	Description
1	Module OK	Indicates the health of the PNC 1 indicates the PNC is functioning properly. 0 indicates the PNC is powering up or has failed.
2	Port1 Link Up	1 indicates the port is connected to another device and is operating correctly. 0 indicates the port is not connected to another device or has an error preventing communications.
3	Port2 Link Up	1 indicates the port is connected to another device and is operating correctly. 0 indicates the port is not connected to another device or has an error preventing communications.
4	Reserved	Reserved. Always 0.
5	Reserved	Reserved. Always 0.
6-8	Reserved	Reserved. Always 0.
9	All Devices Connected*	1 indicates all configured devices are connected and communicating over PROFINET. 0 indicates no devices are configured or one or more configured devices have not established a PROFINET connection.
10-32	Reserved	Set to 0

* Individual device statuses (as reported by the PNIO_DEV_COMM function block) are updated prior to the All Devices Connected bit. Therefore, it is possible (depending on PNC loading) to see via the PNIO_DEV_COMM function block that every individual device is connected while the All Devices Connected bit is not yet set. To avoid this inconsistency, it is recommended that the All Devices Connected bit be checked first, before checking individual device connection status using the PNIO_DEV_COMM function block.

For details on using the PNIO_DEV_COMM function block, refer to "11.7.2, PNIO_DEV_COMM Function Block."

11.8.3 PROFINET IO Alarms

PROFINET IO uses Alarms to transfer indications of changes or problems in the remote IO-Device. For example, Diagnosis Alarms are used to indicate a problem with a channel such as a short circuit, a blown fuse, or overtemperature condition. The PNC translates PROFINET alarms into CPU faults for the RXi CPU. The illustration below shows what happens when a problem is detected on one of the channels of a module within a remote IO-Device, in this case, a VersaMax PROFINET Scanner.



1. The PROFINET Controller establishes Application Relationship (AR) and then Alarm Communication Relationship (CR) with IO-Device
2. Module detects a problem (such as short-circuit) and reports it to IO-Device (PROFINET Scanner).
3. IO-Device sends Alarm to PROFINET Controller via Alarm CR.
4. PROFINET Controller logs fault in the RXi CPU fault table.
5. PROFINET Controller sends Alarm ACK to IO-Device via Alarm CR.

11.8.3.1 PROFINET Alarm Action

If a PROFINET I/O Alarm occurs, the action is Diagnostic (the action type is not configurable). When a fault in this group is processed, the CPU sets the following status bits: #ANY_FLT, #IO_FLT, #IO_PRES, and #PNIO_ALARM (%SA30).

The #PNIO_ALARM status bit is cleared when the I/O fault table is cleared.

11.8.4 PROFINET Controller Faults in the Fault Tables

PNC faults are logged in the CPU's fault tables: the Controller Fault Table or the I/O Fault Table. For summaries of faults and corrective actions, see:

8.4.2, Controller Fault Descriptions and Corrective Actions

8.5.3, I/O Fault Descriptions and Corrective Actions

11.8.4.1 Clearing the RXi Fault Tables

- Clearing one or both of the CPU's fault tables has no effect on the Diagnosis conditions maintained by any IO-Device.
- When the CPU's fault tables are cleared, PROFINET-related faults are not re-reported, even if the condition that is causing the fault still exists. You should address or record the CPU faults before clearing them.

Chapter 12. PROFINET Redundant Media

The PNC supports PROFINET Media Redundancy Protocol (MRP). It can be used as either a Media Redundancy Manager (MRM) or Media Redundancy Client (MRC) on a redundant media ring.

Media Redundancy enables the network to recover from a network link or switch failure. Because it operates transparently to applications using the network, no changes to the application are needed to use media redundancy.

This chapter describes:

- PROFINET Media Redundancy Protocol
- MRP Failover Performance
- Ring Topology with One Controller
- Ring Topology with Multiple Controllers
- Setting Up Media Redundancy Protocol for a PROFINET IO-Controller
- Sequence for Enabling Media Redundancy
- Sequence for Replacing a Media Redundancy Manager
- Procedure for Disabling Media Redundancy

12.1 PROFINET Media Redundancy Protocol

PROFINET Media Redundancy Protocol (MRP) supports devices configured in a ring topology with a maximum of 1 Manager and 63 Clients. It is based on the functions of IEC 62439. Media Redundancy Protocol is not routable between different IP subnets.

Each device within a Redundant Media network has at least two physical pathways to two other devices on the network. To connect to the ring, each device requires an integrated switch with at least two external ports (ring ports) that support Media Redundancy Protocol. Devices that are not MRP-capable can be connected to a device such as an MRP-capable switch in the ring, but they cannot be in the ring themselves. The redundant paths only extend to the devices on the ring that are MRP-capable and enabled.

One of the devices on the ring must be configured as the MRM, and all the other devices must be configured as MRCs.

The MRM disables one of the segments of the ring so that a loop is not created in the network. To disable a segment, the MRM either:

- blocks one of its two Ethernet switch ports used to form the ring when the ring is closed, or
- forwards communications to both switch ports if one of the other ring segments is missing (ring open) and passes messages through the Media Redundancy Manager to communicate with devices on the other side of the failed segment.

When Media Redundancy is enabled for the RXi PROFINET Controller, it can be used as either an MRM or MRC.

12.2 MRP Failover Performance

There are two ways an MRM detects a break in the ring:

- a message from an MRC that provides LinkUp/LinkDown detection
- a test packet timeout interval

A network using Media Redundancy Protocol recovers from a ring failure within 80 milliseconds when running at 100 Mbps full duplex with default values. Actual failover time depends on the device responsiveness to network disconnection and reconnection, number of devices in the ring, media speed, length of media, and frequency of sending test frames over the network. Network recovery time is shorter with fewer devices, faster media speed, and shorter media lengths. Third-party devices in the MRP ring may introduce additional network recovery time.

For bumpless network recovery (without disturbing IO communications to an IO-Device), the Update Rate for the IO-Device should be configured to be greater than 1/3 of the network recovery time. This permits the ring to be disconnected or reconnected without timing out the communication connection between the IO-Device and its IO-Controller. Devices that do not provide LinkUp/LinkDown detection of failure or recovery of a connection should be taken into account when calculating network recovery time.

12.2.1 Guidelines for Update Rates for Bumpless Operation during MRP Ring Recovery

If an application requires the PROFINET IO Devices to operate bumplessly through ring network recovery (no observed loss and subsequent addition of PROFINET IO Devices while the ring network recovers), the following network and application design guidelines for minimum IO Update Rates must be observed:

- If only one RXi is in the ring acting as the Media Redundancy Manager (MRM) and all of the Media Redundancy Clients (MRCs) are VersaMax PNSs:

  Using the PROFINET ports on the RXi, IO Update Rate = 1 ms minimum

- If multiple RXis are in the ring (one RXi acting as the MRM and other RXi(s) as MRC(s)) where VersaMax PNSs are the only PROFINET IO Devices:

  IO Update Rate = 16 ms minimum
  MRP Test Packet Interval = 10 ms
  MRP Test Packet Count to 2.

- If 3rd party MRCs used in the ring, set a minimum IO update rate to the larger of the options that follow:
  - Minimum IO Update Rate configurable in PME that is at least 1/3 the time of the worst-case ring recovery stated by 3rd party manufacturer, regardless of ports utilized. (for example, if a manufacturer states their worst-case ring recovery is 90 ms, the minimum IO update rate allowed would be 90/3 = 30 ms ~ 32ms.)

    or

  - IO Update Rate = 16 ms minimum
    MRP Test Packet Interval = 10 ms
    MRP Test Packet Count = 2.

12.2.2 Effect of MRC LinkUp/LinkDown Detection on Failover Performance

If a client device in a media redundancy ring provides LinkUp/LinkDown detection, the network recovery time can be significantly shorter than the test packet timeout interval because the break is detected immediately.

When MRCs do not provide LinkUp/LinkDown detection, network recovery time also depends upon the *test packet timeout interval* because a break in the ring will not be detected until this interval has elapsed.

12.2.3 Test Packet Timeout Interval

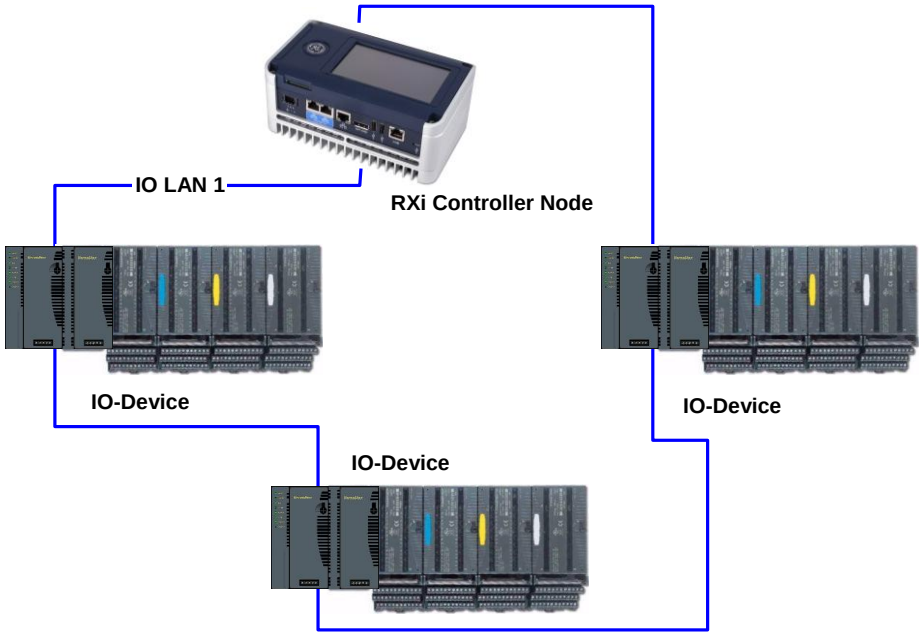
The Default Test Interval and Test Monitoring Count parameters determine the frequency of network integrity checks and the number of failed integrity checks to allow before declaring a ring failure. For a PNC acting as Media Redundancy Manager, these parameters are set as part of the Proficy Machine Edition hardware configuration and determine the *Test Packet Timeout Interval.*

Test Packet Timeout Interval = Default Test Interval × (Test Monitoring Count plus 1)

(For example, the default test packet timeout interval is 20ms × (3+1) = 80ms.)

12.3 Ring Topology with One Controller

The diagram below illustrates a simple ring topology with one RXi Controller node and three IO-Devices on the same PROFINET network. In this example, the RXi Controller is configured to be the Media Redundancy Manager and the IO-Devices, in this case VersaMax PNS modules, are configured to be Media Redundancy Clients.

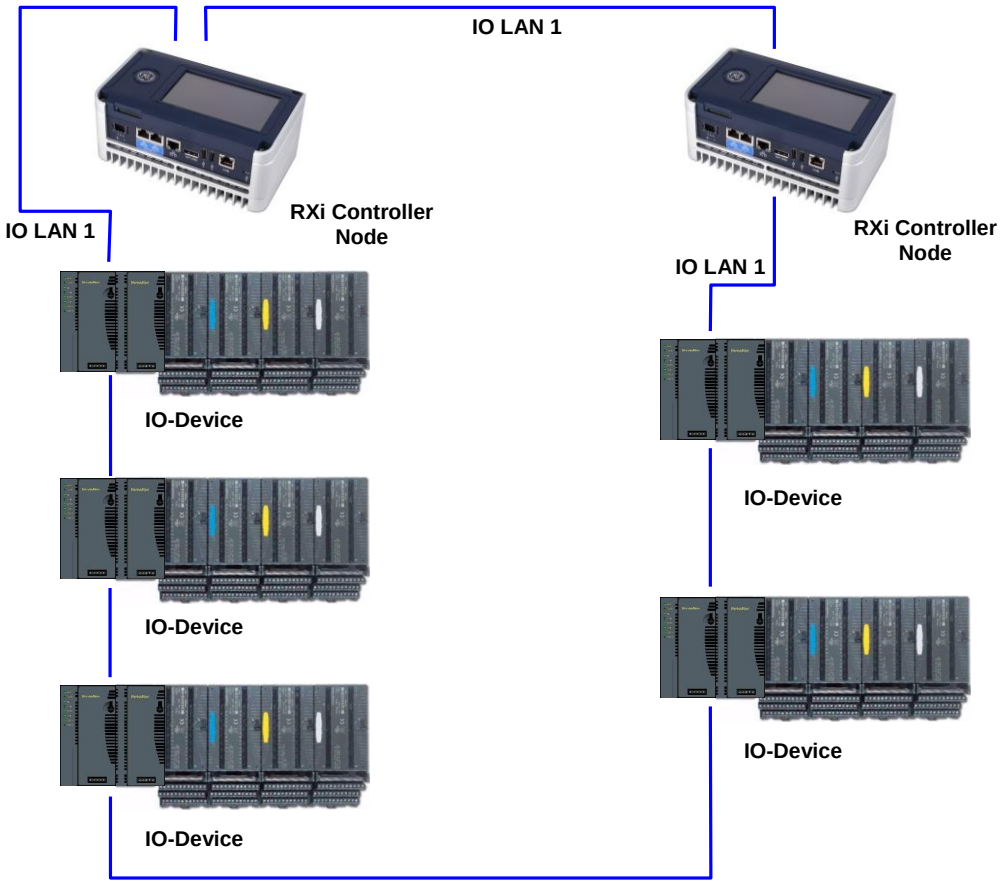


As a Media Redundancy Manager, the PNC detects when the network ring has been broken and repaired. Each time the PNC detects that the ring has been broken or repaired, a fault is reported in in the CPU's I/O Fault Table. As a Media Redundancy Manager, the PNC detects whether another Media Redundancy Manager is on the same ring. If more than one Media Redundancy Manager is present on the same ring, the PNC logs an entry in the I/O Fault Table. Upon detecting that there are no longer multiple Media Redundancy Managers on the ring, the PNC also logs a fault in the I/O Fault Table to indicate that the invalid setup has been resolved.

12.4 Ring Topology with Multiple Controllers

The next illustration shows a more complex network. There are two RXi CPU nodes and five IO-Devices, in this case, VersaMax PNS modules.

All devices are on the same network ring. One PNC is configured as the Media Redundancy Manager (MRM); the other PNC and all PROFINET Scanners are configured as Media Redundancy Clients (MRCs).



12.5 Setting Up Media Redundancy Protocol

Media Redundancy must first be set up in the hardware configuration of all devices on the ring. Parameters for configuring Media Redundancy using Proficy Machine Edition are described in “Media Redundancy Tab” on page 51.

For port locations, refer to 1.2, “RXi Controller User Features.”

12.5.1 Media Redundancy Setup for a PROFINET Controller

A PNC can be configured as either a Media Redundancy Manager (MRM) or a Media Redundancy Client (MRC). Only one device on the ring can be configured as the Media Redundancy Manager. All other devices on the ring must be configured as Media Redundancy Clients.

By default, Media Redundancy is disabled for the PNC. To be used, it must first be enabled and set up in the configuration.

Configuring a Media Redundancy Manager includes specifying a test interval and retry count for ring failure checking. It also includes specifying which of the PNC’s switch ports are connected to the ring.

The PNC stores the current Media Redundancy Protocol configuration settings in non-volatile storage so it can configure the switch port appropriately at powerup. The PNC disables all its external Ethernet ports until it has retrieved and applied the Media Redundancy Protocol configuration from its non-volatile storage.

12.5.2 Sequence for Enabling Media Redundancy

To avoid network loops occurring before the Media Redundancy configuration parameters are stored, the network must first be set up with the ring broken at one point. Otherwise, packets could continuously cycle on the network and use up significant network bandwidth. The ring should not be closed until the Media Redundancy configuration parameters are successfully stored to the Media Redundancy Manager and the Media Redundancy Manager is operational.

If more than one Media Redundancy Manager is present on the same ring, each PNC that is configured as a Media Redundancy Manager will log a fault in the CPU’s I/O Fault Table. When the hardware configuration is cleared, the Media Redundancy configuration parameters (Media Redundancy Manager or Client) are kept until a new setting is stored. If power is lost, the Media Redundancy configuration settings are preserved.

Note: When configuring a Media Redundancy Manager in an open ring, the PNC may initially report the ring to be closed. This indication will immediately be followed by a ring open event. This is expected behavior defined by the MRP specification.

12.6 Sequence for Replacing a Media Redundancy Manager

If it is necessary to replace the PNC that is serving as the Media Redundancy Manager, the replacement module must be set up as a Media Redundancy Manager before adding it to the ring. Alternatively, the ring must be opened before powering up the new PNC and adding it to the network, as described above.

The following procedure can be used to replace a PNC that is operating as Media Redundancy Manager while allowing the devices on the ring to continue operating.

1. Physically disconnect both of the ring's network connections from the RXi Controller that is serving as the Media Redundancy Manager.
2. Replace the RXi Controller.
3. Using the Machine Edition programmer, make sure the hardware configuration is stored to the RXi Controller.
4. Using the Machine Edition programmer, view the Fault Tables to make sure the PNC is operating correctly.
5. Physically reconnect both of the ring's network connections to the replacement RXi Controller.

12.7 Procedure for Disabling Media Redundancy

When disabling Media Redundancy, the ring must be physically opened before storing configuration to the modules. Here is the procedure to successfully disable Media Redundancy on a network.

1. If the ring has no breaks in it, physically disconnect one (and only one) of the ring's network connections from the PNC that's currently the Media Redundancy Manager.
2. Change the configuration for the device that is the Media Redundancy Manager so that it will no longer be the Media Redundancy Manager. If a PNC is the Media Redundancy Manager, use Proficy Machine Edition to disable the Media Redundancy role on that PNC and then to download the hardware configuration. (If a third-party IO-controller is the Media Redundancy Manager, the appropriate third-party configuration tool must be used instead.)

Chapter 13.PROFINET Network Management

The PNC supports the SNMP (Simple Network Management Protocol) and LLDP (Link Layer Discovery Protocol) standards to facilitate network management.

This chapter describes:

- SNMP
  - Overview of SNMP
  - Supported SNMP Features
  - SNMP Read Access
  - MIB-II Groups Supported
  - MIB-II System Group Values
- LLDP
  - Overview of LLDP
  - LLDP Operation
  - LLDP TLV Structures

13.1 SNMP

The PNC's built-in SNMP (Simple Network Management Protocol) Server/Agent function can be used by a third-party SNMP Client or Network Management Station to access network data.

13.1.1 Overview of SNMP

SNMP is a UDP-based network protocol that facilitates the exchange of management information between network devices. An SNMP-managed network consists of three key components: managed devices, agents, and network management systems (NMS).

A managed device is a network node that contains an SNMP agent and that resides on a managed network. Managed devices exchange management information with NMSs via their local agent.

An agent is a network management software module that resides on a managed device. An agent maintains a collection of management information that it provides in an SNMP-compatible format upon request to NMSs.

An NMS executes applications that monitor and control managed devices.

The collections of management information maintained by agents and managed devices are generally referred to as Management Information Bases (MIBs). A MIB is a collection of information organized into a hierarchical structure where related items are grouped together in "groups." MIBs are used to organize and standardize the management information on agents and can be accessed via commands provided within the SNMP protocol.

The SNMP protocol is currently defined by five protocol specifications: SNMPv1, SNMPv2, SNMPv2c, SNMPv2u, and SNMPv3.

NMSs use four basic classes of SNMP commands to retrieve and alter data on managed devices:

- The read (GET) command is used by an NMS to monitor managed devices through examination of the values of different variables maintained within the managed devices' MIBs.
- The trap (TRAP, INFORM) command is used by an NMS to get asynchronous notifications when certain events occur on managed devices. In SNMP Version 2, traps are not set via the protocol but are defined at the managed device by local action.
- Traversal operations (GETNEXT and GETBULK) are used by an NMS to determine which variables a managed device supports and to sequentially gather information from the MIBs.

SNMP is a well-defined protocol; additional information is available in books and the World Wide Web.

13.1.2 Supported SNMP Features

The PNC supports:

- SNMPv1 and SNMPv2c
- MIB-II (see RFC 1213 for details)
- LLDP MIB (see IEEE 802.1AB for details)
- LLDP 802.3 Extension MIB (see IEEE 802.1AB for details)

The PNC does ***not:***

- Support SNMPv3.
- Provide asynchronous notifications via SNMP.
- Generate SNMP TRAP or INFORM messages.

13.1.3 SNMP Read Access

SNMP provides standard commands that allow a client to read SNMP data. The PNC supports all of these commands.

- GET
- GETNEXT
- GETBULK

The SNMP Community strings supported for read access are:

- public
- pub
- icmp
- private
- priv

These community strings are fixed and cannot be changed.

13.1.4 SNMP Write Access

The RXi Controller does not support SNMP writes.

13.1.5 MIB-II Groups Supported

The PNC supports all of the mandatory values in the following groups of the MIB-II specification:

- System group
- Interfaces group
- IP group
- ICMP group
- TCP group
- UDP group
- Transmission group (802.3 Ethernet subset)
- SNMP group

The EGP group and Address Translation group of MIB-II are not supported.

The meanings of the values in these groups are well defined by standard literature. However, the contents of some values are implementation-specific. This section explains contents of values specific to the PNC.

13.1.5.1 Interface Number

Many MIB-II groups use an SNMP interface number to refer to a device port. For the RXi PNC the interface numbers are:

Physical Port	SNMP Interface Number
External Port #1	1
External Port #2	2
Internal Data Port	3

13.1.5.2 MIB-II Interface Group Values

For the MIB-II Interface Group, the *ifDescr* value provides a textual description of the corresponding interface/port. The values returned for that item are the same as the port numbers listed above.

The counters *ifInDiscards* and *ifInUnknownProtocols* are always reported as zero.

13.1.6 MIB-II System Group Values

Many of the items provided by the MIB-II System Group are implementation-specific. The values that are returned for the PNC are described below.

Name	Value
sysDescr	Text description of the device being managed. For example: PROFINET IO-Controller, ICRXICTL000
sysObjectID	Uniquely identifies the kind of device being managed. The first part of the value returned (1.3.6.1.4.1.24893) indicates that this is a GE Intelligent Platforms device. The two numbers after that (1.5) indicate that it is a product made for the Control Systems Business. The three numbers after that identify (in order): ▪ Family: RXi = 1 ▪ Product: PACSystems RXi PNC = 2 ▪ Model: (specific to each Family/Product combination) For the RXi PNC: 1.3.6.1.4.1.24893.1.5.1.2.1
sysContact	Nothing (empty string)
sysName	The Device Name assigned to the node. If the Device Name hasn't been assigned yet, an empty string is returned.
sysLocation	Nothing (empty string)
sysServices	79 (0x4F) This value indicates the PNC has Layers 1–4 (physical, datalink, network and transport) and Layer 7 (application) functionality.

13.2 LLDP

The PNC implements the Link Layer Discovery Protocol (LLDP). A PROFINET IO-Supervisor or other network host may use LLDP to discover the PROFINET network.

13.2.1 Overview of LLDP

The Link Layer Discovery Protocol is an IEEE standardized protocol used by network devices to advertise their identity and capabilities and receive that information from physically adjacent link layer peers. LLDP data packets are sent by devices from each of their interfaces at fixed intervals. The LLDP packets are multicast and are not forwarded by network switches.

Each LLDP packet consists of one LLDP Data Unit (LLDPDU). Each LLDPDU is a sequence of Type-Length-Value (TLV) structures.

LLDP Data Unit Contents

Chassis ID TLV structure (mandatory, always first)
Port ID TLV structure (mandatory, always second)
Time to Live TLV structure (mandatory, always third)
Optional Structures (any number of these, in any order) . . .
End of PDU TLV (mandatory, always last)

13.2.2 LLDP Operation

The LLDP agent in the PNC multicasts LLDP messages to the network at 5 second intervals, or when a change occurs in any of the local data that is delivered within the LLDP message. The LLDP messages are transmitted from each external port that is connected to a network.

13.2.3 LLDP TLVs

The following is a brief description of the RXi PNC LLDP TLVs.

13.2.3.1 Chassis ID TLV

The Chassis ID is always the first TLV in the LLDPDU. Chassis ID identifies the particular device on the network. PROFINET defines two possible Chassis ID formats; each format has a different Chassis ID subtype. The PNC generally uses the Name of Station (NOS) as the Chassis ID. If for some reason the station name is changed to the empty string “”, the internal MAC address is used instead.

ID Subtype	Meaning	Value
7	PNIO Chassis ID	Name of Station (character string)
4	MAC Address	Internal MAC Address of Ethernet controller

13.2.3.2 Port ID TLV

The Port ID is always the second TLV in the LLDPDU. The Port ID identifies the individual network port on the device. PROFINET defines the Port ID as a character string.

ID Subtype	Meaning	Value
7	PNIO Port ID	“port-PPP-SSSSS” (length = 14 bytes)

PPP specifies the decimal port number in the range 001 to 002. SSSSS specifies the decimal slot number of the Controller in the range 00000 to 65535.

13.2.3.3 Time to Live TLV

Time to Live is always the third TLV in the LLDPDU. Time to Live specifies the number of seconds that the information in this LLDPDU remains valid after delivery on the network. A Time to Live value of 0 instructs the receiver to immediately invalidate the data in this LLDPDU and is issued when the PNC has changed a parameter that is advertised in its LLDP.

The PNC sets the Time to Live value to 20 seconds.

13.2.3.4 End Of PDU TLV

The End Of PDU TLV is always the last TLV in the LLDPDU. This TLV carries no device information.

13.2.3.5 PROFINET Port Status TLV

The PROFINET Port Status TLV indicates the current PROFINET status of the network port over which this LLDP message is sent. As the PNC does not support RT Class 2 or RT Class 3, the subfields are set to off (0x0000).

13.2.3.6 PROFINET MRP Port Status TLV

The PROFINET MRP Port Status indicates the current MRP status of the network port over which this LLDP message is sent.

This TLV is required when the LLDP sender is an MRP port. It is not present in the LLDP packet issued from a non-MRP port.

The PNC supports the following values:

Subfield	Value
MRP Domain ID	Contains the MRP domain UUID. Assigned internally by the network node. Set to 0xFFFFFFFF-FFFF-FFFF-FFFF- FFFFFFFFFFFF.
MRRT Port Status	0x00 = Off / Not Used

13.2.3.7 PROFINET Chassis MAC TLV

The PROFINET Chassis MAC indicates the internal MAC address used by the PROFINET stack. This is not the MAC address of any individual network port.

13.2.3.8 MAC/PHY Config/Status TLV

The MAC/PHY Configuration/Status indicates the supported auto-negotiation capability, the current negotiated capability, and the MAU type of the network port over which this LLDP message is sent.

Field	Values	RXi PNC
Auto-negotiation Support/Status	Bit 0: Autonegotiation supported (1 = Yes, 0 = No) Bit 1: Autonegotiation status (1 = Enabled, 0 = Disabled) Bits 2-7: Reserved	0x03 (Autonegotiation supported and enabled)
Auto-negotiation Advertised Capability		Copper Ports: 0x6C03 = 10/100/1000T HD+FD
Operational MAU Type	Typical values are: 0x000B: 10BaseT FD 0x0010: 100BaseTX FD 0x001E: 1000BaseT FD (Copper)	Copper Ports: 0x001E = 1000T FD

13.2.3.9 Management Address TLV

The Management Address is the address associated with the local LLDP agent that may be used to reach higher layers within the device to assist with LLDP operations. Typically, this indicates how to reach the LLDP MIB. The Management Address TLV contains three fields: Management Address, Interface Number, and Object Identifier (OID).

Management Address

The Management Address indicates the management address of this device. For the RXi PNC, the management address is the IP address of the device.

Subtype	Meaning	Address Value
4	IPv4 address	IPv4 address of this device

Interface Number

The Interface Number identifies a specific interface or port at the specified management address. Thus the interface number varies with the network port number. The RXi Controller uses the current network port number.

Subtype	Meaning	Address Value
3	System port number	Management port number (1-max network ports)

Object Identifier (OID)

The OID specifies a MIB object (typically the LLDP MIB) in ASN.1 format that is reachable at the specified management address and interface number. A length of zero means that an OID is not provided. The OID value is internally assigned by, and meaningful only to, the LLDP device.

Appendix A CPU Performance Data

This appendix contains instruction and overhead timing collected for the RXi CPU. This timing information can be used to predict CPU sweep times.

A.1. Instruction Timing

The tables in this section list the execution and incremental times in microseconds for each function supported by the RXi CPU.

Execution Times

Two execution times are shown for each instruction.

Execution Time	Description
Enabled	Time in microseconds required to execute the function or function block when power flows into the function with valid inputs.
Disabled	Time in microseconds required to execute the function when it is not enabled.

Notes:

- All times represent typical execution time. Times may vary with input and error conditions.
- Enabled time is for single length units of word-oriented memory.
- DO_IO time was measured using a discrete output module.
- Timers are updated each time they are encountered in the logic by the amount of time consumed by the last sweep.

RXi Incremental Times

An Increment time is shown for functions that can have variable length inputs.

Incremental time is added to the base function time for each addition to the length of an input parameter. This time applies only to functions that can have varying input lengths (Search, Array Moves, etc.)

Units:

- For table functions, increment is in units of length specified.
- For bit operation functions, increment is microseconds per bit.
- For data move functions, microseconds per unit.

RXi Instruction Times

Instruction	Enabled	Disabled	Increment
ABS_DINT	0.464	0.296	0
ABS_INT	0.457	0.297	0
ABS_LREAL	0.574	0.3	0
ABS_REAL	0.455	0.278	0
ACOS	0.725	0.272	0
ACOS_LREAL	0.972	0.293	0
ADD_DINT	0.598	0.414	0
ADD_INT	0.548	0.416	0
ADD_LREAL	0.701	0.429	0
ADD_REAL	0.581	0.411	0
ADD_UINT	0.563	0.402	0
AND_DWORD	0.868	0.509	0.04076
AND_WORD	0.856	0.512	0.02306
ARRAY_MOV_BIT	1.021	0.651	0.00313
ARRAY_MOV_BYTE	0.839	0.645	0.00292
ARRAY_MOV_DINT	0.842	0.642	0.01374
ARRAY_MOV_DWORD	0.85	0.643	0.01432
ARRAY_MOV_INT	0.876	0.662	0.00613
ARRAY_MOV_UINT	0.887	0.658	0.00596
ARRAY_MOV_WORD	0.858	0.665	0.00646
ARRAY_RANGE_DINT	1.036	0.589	0.09073
ARRAY_RANGE_DWORD	1.044	0.598	0.0912
ARRAY_RANGE_INT	1.04	0.573	0.0929
ARRAY_RANGE_UINT	1.044	0.593	0.1036
ARRAY_RANGE_WORD	1.02	0.597	0.1046
ASIN	0.725	0.272	0
ASIN_LREAL	0.874	0.275	0
ATAN	0.648	0.285	0
ATAN_LREAL	0.741	0.295	0
BCD4_TO_INT	0.48	0.334	0
BCD4_TO_REAL	0.499	0.33	0
BCD4_TO_UINT	0.506	0.356	0
BCD8_TO_DINT	0.492	0.298	0
BCD8_TO_REAL	0.526	0.317	0
BCLR_DWORD	0.608	0.343	0
BCLR_WORD	0.628	0.362	0
BIT_POS_DWORD	1.05	0.399	0.45679
BIT_POS_WORD	0.836	0.421	0.20669
BIT_SEQ	0.088	0.087	0
BLK_CLR_WORD	0.56	0.33	0.00707
BLKMOV_DINT	0.898	0.694	0
BLKMOV_DWORD	0.888	0.683	0

Instruction	Enabled	Disabled	Increment
BLKMOV_INT	0.868	0.692	0
BLKMOV_REAL	0.863	0.65	0
BLKMOV_UINT	0.83	0.657	0
BLKMOV_WORD	0.862	0.699	0
BIT_SET_DWORD	0.621	0.365	0
BIT_SET_WORD	0.621	0.371	0
BIT_TST_DWORD	0.732	0.429	0.0002
BIT_TST_WORD	0.72	0.425	0.0004
CALL_C	2.154	0.082	0
CALL_LDBK	1.649	0.102	0
CALL_PSB	1.203	0.096	0
CMP_DINT	0.642	0.301	0
CMP_INT	0.624	0.289	0
CMP_LREAL	0.754	0.352	0
CMP_REAL	0.661	0.302	0
CMP_UINT	0.655	0.312	0
COMM_REQ	248.372	0.476	0
COS	0.603	0.274	0
COS_LREAL	0.72	0.294	0
DATA_INIT_ASCII	0.21	0.366	0.00339
DATA_INIT_COMM	0.222	0.335	0.00607
DATA_INIT_DINT	0.21	0.343	0.01236
DATA_INIT_DLAN	-	-	-
DATA_INIT_DWORD	0.199	0.323	0.01262
DATA_INIT_INT	0.226	0.339	0.00612
DATA_INIT_LREAL	0.254	0.378	0.02637
DATA_INIT_REAL	0.221	0.328	0.01241
DATA_INIT_UINT	0.224	0.336	0.00622
DATA_INIT_WORD	0.225	0.323	0.00648
DEG_2_RAD	0.451	0.265	0
DEG_2_RAD_LREAL	0.597	0.294	0
DINT_TO_BCD8	0.549	0.308	0
DINT_TO_INT	0.442	0.301	0
DINT_TO_LREAL	0.449	0.304	0
DINT_TO_REAL	0.498	0.304	0
DINT_TO_UINT	0.459	0.298	0
DIV_DINT	0.624	0.436	0
DIV_INT	0.578	0.413	0
DIV_LREAL	0.731	0.41	0
DIV_MIXED	0.569	0.404	0
DIV_REAL	0.585	0.399	0
DNCTR	1.044	1.041	0

Instruction	Enabled	Disabled	Increment
DO_IO	4.79	0.405	0
DO_IO WITH ALT	4.882	0.412	0
DRUM	1.543	1.227	0
EQ_DATA	-	-	-
EQ_DINT	0.51	0.256	0
EQ_INT	0.544	0.255	0
EQ_LREAL	0.469	0.3	0
EQ_REAL	0.538	0.272	0
EQ_UINT	0.481	0.234	0
EXP	0.645	0.285	0
EXP_LREAL	0.883	0.311	0
EXPT	1.019	0.344	0
EXPT_LREAL	0.667	0.346	0
FIFO_RD_DINT	0.799	0.492	0.01372
FIFO_RD_INT	0.817	0.495	0.00614
FIFO_WRT_DINT	0.744	0.477	0.00003
FIFO_WRT_INT	0.763	0.496	-0.00017
FOR	0.434	0.302	0
GE_DINT	0.511	0.268	0
GE_INT	0.539	0.249	0
GE_LREAL	0.602	0.297	0
GE_REAL	0.557	0.261	0
GE_UINT	0.507	0.26	0
GT_DINT	0.46	0.252	0
GT_INT	0.525	0.273	0
GT_LREAL	0.481	0.329	0
GT_REAL	0.472	0.248	0
GT_UINT	0.462	0.233	0
INT_TO_BCD4	0.495	0.306	0
INT_TO_DINT	0.429	0.293	0
INT_TO_REAL	0.472	0.32	0
INT_TO_UINT	0.467	0.306	0
JUMPN	0.134	0.056	0
LE_DINT	0.492	0.238	0
LE_INT	0.553	0.262	0
LE_LREAL	0.478	0.312	0
LE_REAL	0.5	0.236	0
LE_UINT	0.506	0.246	0
LIFO_RD_DINT	0.73	0.485	0.00017
LIFO_RD_INT	0.745	0.481	-0.00011
LIFO_WRT_DINT	0.762	0.512	-0.00003
LIFO_WRT_DWORD	0.762	0.498	-0.00027

Instruction	Enabled	Disabled	Increment
LIFO_WRT_INT	0.74	0.479	0.00014
LN	0.644	0.282	0
LN_LREAL	0.728	0.289	0
LOG	0.62	0.264	0
LOG_LREAL	0.719	0.27	0
LREAL_TO_DINT	0.677	0.3	0
LREAL_TO_REAL	0.497	0.263	0
LT_DINT	0.49	0.278	0
LT_INT	0.515	0.258	0
LT_LREAL	0.447	0.313	0
LT_REAL	0.485	0.258	0
LT_UINT	0.45	0.231	0
MASK_CMP_DWORD	1.408	0.779	0.05294
MASK_CMP_WORD	1.364	0.776	0.04576
MCRN	0.166	0.159	0
MOD_DINT	0.618	0.42	0
MOD_INT	0.585	0.418	0
MOD_UINT	0.585	0.42	0
MOVE_FROM_FLAT	2.256	1.123	0
MOVE_TO_FLAT	2.256	1.123	0
MOVE_BIT	0.761	0.387	0.00249
MOVE_DATA	-	-	-
MOVE_DATA_EX_INPUTREF	2.32	1.076	0
MOVE_DATA_INPUTREF	1.942	0.847	0
MOVE_DINT	0.526	0.327	0.01379
MOVE_DWORD	0.551	0.374	0.01347
MOVE_INT	-	-	0
MOVE_LREAL	0.583	0.383	0.0287
MOVE_REAL	0.529	0.354	0.0134
MOVE_UINT	0.541	0.347	0.0062
MOVE_WORD	0.549	0.374	0.006
MUL_DINT	0.633	0.418	0
MUL_INT	0.575	0.398	0
MUL_LREAL	0.705	0.425	0
MUL_MIXED	0.589	0.457	0
MUL_REAL	0.595	0.418	0
MUL_UINT	0.544	0.403	0
NE_DINT	0.496	0.289	0
NE_INT	0.466	0.277	0
NE_LREAL	0.454	0.319	0
NE_REAL	0.49	0.269	0
NE_UINT	0.482	0.265	0

Instruction	Enabled	Disabled	Increment
NOT_DWORD	0.707	0.362	0.02917
NOT_WORD	0.699	0.371	0.01689
OFDTR	1.104	1.02	0
ONDTR	1.141	0.919	0
OR_DWORD	0.905	0.532	0.04118
OR_WORD	0.855	0.512	0.02337
PID_IND	1.692	1.563	0
PID_ISA	1.727	1.594	0
RAD_2_DEG	0.448	0.268	0
RAD_2_DEG_LREAL	0.597	0.292	0
RANGE_DINT	0.742	0.542	0
RANGE_DWORD	0.757	0.552	0
RANGE_INT	0.715	0.562	0
REAL_TO_DINT	0.54	0.291	0
REAL_TO_INT	0.586	0.303	0
REAL_TO_LREAL	0.491	0.264	0
REAL_TO_UINT	0.554	0.291	0
ROL_DWORD	0.813	0.393	0.03971
ROL_WORD	0.805	0.4	0.02681
ROR_DWORD	0.793	0.374	0.0322
ROR_WORD	0.784	0.375	0.02617
SCALE_DINT	0.942	0.573	0
SCALE_INT	0.967	0.583	0
SCALE_UINT	0.942	0.552	0
SCANSET_IO_INOUT	32.943	0.423	0
SHFR_BIT	1.282	0.68	0.0053
SHFR_DWORD	1.578	1.087	0.04544
SHFR_WORD	1.622	1.125	0.03731
SHL_DWORD	1.07	0.769	0.0328
SHL_WORD	1.127	0.785	0.02066
SHR_DWORD	1.109	0.767	0.04167
SHR_WORD	1.076	0.752	0.02627
SIN	0.621	0.277	0
SIN_LREAL	0.743	0.297	0
SORT_INT	9.27	0.339	0.19969
SORT_UINT	9.214	0.336	0.2004
SORT_WORD	9.208	0.332	0.1995
SQRT_DINT	0.525	0.283	0
SQRT_INT	0.478	0.289	0
SQRT_LREAL	0.738	0.302	0
SQRT_REAL	0.551	0.301	0
SRCHBYTE	0.9	0.6	0.01154

Instruction	Enabled	Disabled	Increment
SRCHDWORD	0.931	0.61	0.01717
SRCHWORD	0.98	0.611	0.01627
SUB_DINT	0.582	0.398	0
SUB_INT	0.556	0.395	0
SUB_LREAL	0.701	0.428	0
SUB_REAL	0.585	0.415	0
SUS_IO	1.309	0.103	0
SVC_REQ 1	1.437	0.244	0
SVC_REQ 2	1.747	0.234	0
SVC_REQ 3	1.018	0.26	0
SVC_REQ 4	0.995	0.242	0
SVC_REQ 5	1.001	0.236	0
SVC_REQ 6	1.013	0.245	0
SVC_REQ 7	2.534	0.231	0
SVC_REQ 8	4.307	0.234	0
SVC_REQ 9	1.441	0.232	0
SVC_REQ 10	1.493	0.234	0
SVC_REQ 11	1.068	0.238	0
SVC_REQ 12	0.634	0.232	0
SVC_REQ 13	1.684	0.259	0
SVC_REQ 14	113.149	0.243	0
SVC_REQ 15	0.626	0.35	0
SVC_REQ 16	1.372	0.237	0
SVC_REQ 17	0.774	0.237	0
SVC_REQ 18	34.532	0.257	0
SVC_REQ 19	1.755	0.256	0
SVC_REQ 20	5.241	0.254	0
SVC_REQ 21	11.909	0.232	0
SVC_REQ 22	0.672	0.238	0
SVC_REQ 23	24.311	0.214	0
SVC_REQ 24	0.611	0.234	0
SVC_REQ 25	0.711	0.23	0
SVC_REQ 50	1.344	0.215	0
SVC_REQ 51	1.409	0.214	0
SW_POS	0.506	0.292	0
SWAP_DWORD	0.623	0.365	0.02767
SWAP_WORD	0.626	0.357	0.016
TAN	0.73	0.276	0
TAN_LREAL	0.855	0.296	0
TBL_RD_DINT	0.77	0.549	0.0002
TBL_RD_INT	0.809	0.585	0.00018
TBL_WRT_DINT	0.894	0.575	0.00012

Instruction	Enabled	Disabled	Increment
TBL_WRT_INT	0.876	0.537	0.00007
TMR	1.11	1.008	0
TOF	1.954	1.059	0
TON	1.177	1.057	0
TP	1.404	1.061	0
TRUNC_DINT	0.591	0.249	0
TRUNC_INT	0.55	0.212	0
UINT_TO_BCD4	0.49	0.313	0
UINT_TO_DINT	0.487	0.346	0
UINT_TO_INT	0.452	0.314	0
UINT_TO_REAL	0.475	0.321	0
UPCTR	1.049	1.042	0
XOR_DWORD	0.88	0.522	0.04049
XOR_WORD	0.887	0.533	0.02306

A.1. PROFINET Controller and PROFINET IO Sweep Impact

The controller CPU sweep impact for a PROFINET IO network is a function of the number of PNCs, the number of PROFINET devices, and the number of each PROFINET device's IO modules. PROFINET IO Performance Examples are provided in Appendix B.

Appendix B PROFINET IO Performance Examples

Note: The configurations and performance numbers shown here are examples; Actual performance timings will vary depending on your exact configuration and network setup.

This section presents various PROFINET IO systems and their measured I/O performance for a simple 16 point Discrete I/O Loopback. Additional details of each system configuration are in a section following the table below.

The I/O Loopback measurements were done by wiring the outputs of a discrete output module directly to the inputs of a discrete input module, and using Ladder Logic to measure the time difference between setting the outputs and reading the same data on the corresponding inputs. Proficy Machine Edition was attached to the RXi during testing. The average, minimum, and maximum loopback times were captured from a 4 hour sample period. In addition, since the CPU sweep time is a primary factor in the accuracy and precision of the I/O Loopback times recorded, the average and maximum CPU sweep times for each system are given.

Note: CPU Sweep time variations between systems are not completely a function of varying PROFINET IO. Sweep time was also influenced by variations in logic.

System Configuration	*# PROFINET Devices/Modules*	*# PROFINET Discretes/Analogs*	*CPU Sweep (ms) Average (Max)*	*16pt Loopback (ms) Average (Min, Max)*
1ms PROFINET Update Rate				
A.1 - Single Device	1 / 4	192 / 0	3.7 (5.8)	9 (6, 18)
A.2 - Multi-Device	8 / 14	824 / 0	5.4 (8.5)	10 (5, 18)
A.2.1 - Multi-Device w/ MRP	8 / 14	824 / 0	5.5 (8.8)	11 (5, 21)
16ms PROFINET Update Rate				
B.1 - Multi-Device	20 / 29	1928 / 0	5.4 (7.9)	21 (5, 43)
B.1.1 - Multi-Device w/ MRP	20 / 29	1928 / 0	5.2 (7.5)	22 (5, 46)
B.2 - Large System	64 / 89	5176 / 445	12.3 (15.6)	35 (11, 51)
B.3 - Very Large System	128 / 275	8376 / 565	19.1 (22.9)	43 (18, 63)

Number of modules only includes I/O modules; does not include head-ends or power supplies.

B.1. System Descriptions

B.1.1. 1ms PROFINET Update Rate Systems

A.1 – Single Device

The PNC was connected to a single VersaMax PNS (IC200PNS001) configured for a 1msec. PROFINET update rate. The PNS contained four mixed discrete 16 point in/out modules (IC200MDD844). Each I/O module had its outputs tied to the inputs of an adjoining I/O module, and each input module was configured with a 1ms Input DC Filter time. The I/O Loopback measurement was taken using one of the IC200MDD844 modules.

A.2 – Multi-Device

The PNC was connected in a network bus configuration to eight different VersaMax PNS modules (IC200PNS001), each configured for a 1ms PROFINET update rate. Each PNS contained at least one discrete input or output module. The I/O Loopback measurement was taken using a 16 point output module (IC200MDL741) owned by one VersaMax PNS, that was then tied to the 16 point input module (IC200MDL640) owned by a different VersaMax PNS. The input module (IC200MDL640) was configured for a 0ms Input DC Filter time.

A.2.1 - Multi-Device with MRP

This system is identical to the *A.2 – Multi-Device* configuration, except that Media Redundancy Protocol (MRP) was in use. To exercise MRP, the network was set up in a ring configuration and the PNC was configured as an MRP Manager with a configured Default Test Interval of 20ms.

B.1.2. 16ms PROFINET Update Rate Systems

B.1 – Multi-Device	This system contained an RXi controller connected in a network bus configuration to 20 VersaMax PNS modules (IC200PNS001), each configured for a 16ms PROFINET update rate. Each PNS contained at least 1 discrete input, discrete output, analog input, or analog output module. The IO Loopback measurement was taken using a 16-point output module (IC200MDL741) owned by one VersaMax PNS, that was then tied to the 16 point input module (IC200MDL640) owned by a different VersaMax PNS. The input module (IC200MDL640) was configured for a 0ms Input DC Filter time.
B.1.1 – Multi-Device with MRP	This system is identical to the B.1 – Multi-Device configuration, except that MRP was in use. The network was set up in a ring configuration and the PNC was configured as an MRP Manager with a configured Default Test Interval of 20ms.
B.2 – Large System	This system contained an RXi controller connected in a network bus configuration to 64 VersaMax PNS modules (IC200PNS001), each configured for a 16ms PROFINET update rate. Each PNS contained at least one discrete input, discrete output, analog input, or analog output module. The I/O Loopback measurement was taken using a 16 point output module (IC200MDL741) owned by one VersaMax PNS, that was then tied to the 16 point input module (IC200MDL640) owned by a different VersaMax PNS. The input module (IC200MDL640) was configured for a 1ms Input DC Filter time.
B.3 – Very Large System	This system contained an RXi controller connected in a network bus configuration to 64 VersaMax PNS modules (IC200PNS001) and 64 third-party PROFINET devices. Each PROFINET device was configured for a 16ms PROFINET update rate and contained at least one discrete input, discrete output, analog input, or analog output module. The I/O Loopback measurement was taken using a 16 point output module (IC200MDL741) owned by one VersaMax PNS, that was then tied to the 16 point input module (IC200MDL640) owned by a different VersaMax PNS. The input module (IC200MDL640) was configured for a 1ms Input DC Filter time.

Appendix C User Memory Allocation

User Memory Size is the number of bytes of memory available to the user for controller applications.

Model	User Memory Size	Bytes
ICRXICTL000	10MB	10,485,760

C.1. Items that Count Against User Memory

The following items count against the CPU memory and can be used to estimate the minimum amount of memory required for an application. Additional space may be required for items such as zipped source files, user heap, and published symbols.

Register Memory Size (%R)	Bytes = %R references configured × 2
Word Memory Size (%W)	Bytes = %W references configured × 2
Analog Inputs (%AI)	If point faults enabled: Bytes = %AI references configured × 3 If point faults disabled: Bytes = %AI references configured × 2
Analog Outputs (%AQ)	If point faults enabled: Bytes = %AQ references configured × 3 If point faults disabled: Bytes = %AQ references configured × 2
Discrete Point Faults	If point faults enabled: Bytes = 3072
Managed Memory (Symbolic Variable and I/O Variable Storage)	The total number of bytes required for symbolic and I/O variables. Calculated as follows: [(number of symbolic discrete bits) × 3 / (8 bits/byte)] + [(number of I/O discrete bits) × Md / (8 bits/byte)] + [(number of symbolic words) × (2 bytes/word)] + [(number of I/O words) × (Mw bytes/word)] Md = 3 or 4. The number of bits is multiplied by 3 to keep track of the force, transition, and value of each bit. If point faults are enabled, the number of I/O discrete bits is multiplied by 4. Mw = 2 or 3. There are two 8-bit bytes per 16-bit word. If point faults are enabled, the number of bytes is multiplied by 3 because each I/O word requires an extra byte.
I/O Scan Set File (included in HWC)	Based on number of scan sets used. **Note:** 32 bytes of user memory are consumed if the application scans all I/O every sweep (the default).
User Programs	See “User Program Memory Usage” page 268 for details on user programs.

C.2. User Program Memory Usage

Space required for user logic includes the following items.

C.2.1. %L and %P Program Memory

%L and %P are charged against your user space and sized depending on their use in your applications. The maximum size of %L or %P is 8192 words per block.

The %L and %P tables are sized to allow extra space for Run Mode Stores according to the following rules.

- If %L memory is not used in the block, the %L memory size is 0 bytes. If %L memory is used in the block, a buffer is added beyond the highest %L address actually used in logic or in the variable table. The default buffer size is 256 bytes, but can be changed by editing the Extra Local Words parameter in the block Properties.
- The same rules apply for the size of %P memory, but %P memory can be used in any block in the program.
- The buffer cannot make the %P or %L table exceed the maximum size of 8,192 words. In such a case, a smaller buffer is used.
- You can add, change, or delete %L and/or %P variables in your application and Run Mode Store the application if these variables fit in the size of the last-stored %L/%P tables (where the "size" includes the previous buffer space), or if going from a zero to non-zero size.
- The size of the %L/%P tables is always recalculated for Stop Mode Stores.

C.2.2. Program Logic and Overhead

The data area for C (.gefelf) blocks is considered part of the user program and counts against the user program size. Additional space is required for information internal to the CPU that is used for execution of the C block.

The program block is based on overhead for the block itself plus the logic and register data being used (that is, %L).

Note: The LD program's stack is not counted against the CPU's memory size.

Note: If your application needs more space for LD logic, consider changing some %P or %L references to %R, %W, %AI, or %AQ. Such changes require a recompilation of the program block and a Stop Mode store to the CPU.

Appendix D Product Certifications and Standards

This appendix describes the compliance markings that appear on the PACSystems RXi Controller and the corresponding standards to which the products have been certified.

D.1. Agency Approvals

Description	Agency Standard or Marking	Comments
N.A. Safety for Information Technology Equipment	c UL us LISTED	Certification by Underwriter's Laboratories to UL 60950-1and CSA 60950-1
N.A. Safety for Industrial Control Equipment	c UL us LISTED	Certification by Underwriter's Laboratories to UL 508 and CSA C22.2 No 142 - M1987
Electromagnetic Compatibility Directive European EMC	CE	Self-Declaration in accordance with European Directives; Refer to Technical Support at http://www.ge-ip.com/ for EC Declaration of Conformity

Note: The agency approvals listed above and on the Declaration of Conformities are believed to be accurate; however, a product's agency approvals should be verified by the marking on the unit itself.

D.2. Standards Overview

For environmental specifications, refer to 1.1.4, Environmental Specifications.

EMC Emissions and Immunity Specifications

EMC Product Standards		
Industrial Control Equipment	**EN 61131-2**	EMC Zone B requirements
Information Technology Equipment	**EN 55024** **EN 55024**	
Generic Standards	**EN 61000-6-2** **EN 61000-6-4**	Immunity Emissions
EMC EMISSIONS		
Emissions	**CISPR 11/EN 55011/ EN55016-2-3**	30 – 1000 MHz, radiated, Group 1, Class A,
	FCC 47CFR15	30 – 7500 MHz, radiated, Class A
	CISPR 22/EN55022	30 – 6000 MHz, radiated, Class A, Ethernet port: 0.15 – 30 MHz, conducted, Class A
EMC IMMUNITY		
Electrostatic Discharge	**EN61000-4-2**[1]	±8.0 kV air, ±4.0 kV contact,
RF Susceptibility	**EN 61000-4-3**[1]	1kHz sine wave, 80% AM 10 V/m, 80 – 1000 MHz 3 V/m, 1.0 – 2.0 GHz 1 V/m, 2.0 – 2.7 GHz
Fast Transient	**EN 61000-4-4**[1]	DC Input: ±2 kV, 33 nF direct Ethernet port, shielded: ±1.1 kV, capacitive clamp
		RS-232 port, unshielded: ±1.1 kV, capacitive clamp
Surge	**EN 61000-4-5**[1]	DC Input: 0.5 kV, 12 ohms (CM); 0.5 kV, 2 ohms (DM) Ethernet port, shielded: ±1.0 kV, 2 ohms (CM)
Conducted RF Immunity	**EN 61000-4-6**[1]	DC Input: 10 V, CDN Ethernet port, shielded: 10 V, current clamp
		RS-232 port, unshielded: 10 V, current clamp
Damped Oscillatory Wave	**EN 61000-4-18**[1]	DC Input: ±2. 5kV, 1 MHz (CM/DM); Ethernet port, shielded: ±2.5 kV, 1 MHz (CM)
		RS-232 port, unshielded: ±2.5 kV, 1 MHz (CM)

[1] EN 61000-4-x series of tests are technically equivalent to the IEC 61000-4-x series.

Product Safety

Ordinary Locations:	
UL/CSA/EN/IEC 60950-1	Requires: LPS or Class 2 power source, Pollution Degree 2 environment
UL508, CSA 22.2 No 142	Requires: LPS, Class 2, or LVLC power source, Pollution Degree 2 environment

D.3. Government Regulations

The FCC requires the following note to be published according to FCC guidelines:

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Industry Canada requires the following note to be published:

Note: This Class A digital apparatus complies with Canadian ICES-003.

@
indirect references, 119
Access control, 46
Air flow, 17
Alarm contacts, 146
Alarm CR, 228
Altitude specification, 14
Analog input register references (%AI), 119
Analog output register references (%AQ), 119
Application relationship (AR), 228
PROFINET
defined, 225
Arrays, 116
accessing elements with variable index, 116
Auto-Located symbolic variables, 113
Autonegotiation, 222
Basic System, 223
Battery, real time clock (RTC), 30
Bit in Word references, 120
Bit references, 121
Blocks
and local data, 94
external, 102
parameterized, 95
program, 94
types, 93
UDFBs, 98
Broadcast
PROFINET, defined, 225
Bulk memory, 119
Cables
Gigabit Ethernet (GbE), 27
summary, 26
Channel Commands, 180
Establish Write Channel (2004), 189
Establishing a channel (3001), 182
Mask Write Register Request to a Modbus Server Device (3009), 195
Open a Modbus TCP Client Connection (3000), 180
Read Data from a Modbus TCP Device (3003), 183
Read/Write Multiple Registers to/from a Modbus Server Device (3005), 193
Channel Error bit, 203, 216
Channel Status bits, 176
Channels
Monitoring, 203
Clearances, 20
Client/Server Capability, 169
Clocks, 83
elapsed time clock, 83
reading with SVCREQ #16 or #50, 83
time-of-day clock, 83
reading and setting, 83
Coherency, 233
COMM_REQs
Channel Commands, 175
Command Block, 175, 179
controlling execution, 176
fault errors, 217
function block, 175
functions, maximum pending, 203
Status words, 176, 177, 203
major and minor error codes, 204
Communication relationship (CR), 228
PROFINET, defined, 225
Configuration
PROFINET update rate, 234
storing (downloading), 70
system, 90
Connectors
Gigabit Ethernet (GbE), 27
PROFINET, 29
Controller Fault Table, 212
Convenience references. See System status references
Coupled variables, 114
CPU memory validation, 90
CPU Node
PROFINET, defined, 225
CPU sweep
Stop modes, 79
Data coherency, 233
Data coherency in communications windows, 77
Data retentiveness

power cycle, 90
Stop to Run mode transition, 123
Data scope, 124
Data Transfer bit, 216
Data types, 128
DCP. *See* Discovery and Configuration Protocol
Declaration of Conformities, 269
Determining if an IP address has been used, 69
Device access point (DAP)
PROFINET, defined, 225
Device Name
PNC, 53
Diagnostics
CPU, 136
Ethernet interface, 212
PNC, 239
Dimensions
mounting, 20
Disabling Media Redundancy, 248
Discovery and Configuration Protocol, 65
PROFINET diagnostics, 164
Discrete references, 121
size and default, 122
Do I/O
in an interrupt block, 111
DO I/O for Remote IO Modules, 237
Downloading configuration
details, 70
Duplicate controller IP addresses, 232
Duplicate device IP addresses, 231, 232
Elapsed time clock, 83
reading with SVCREQ #16 or #50, 83
Enabling Media Redundancy, 247
Enhanced security, 46
Environmental specifications, 14
Errors
in floating point numbers, 131
Ethernet (GbE)
connectors, 27
Ethernet (GbE) interface
configuring, 68
features, 169
operation, 169
Ethernet interface status bits, 215
Examples
alarm contacts, 146
bit in word references, 120
configuring submodules on the PROFINET network, 62
fault classes, 136
PNIO_DEV_COMM, 236
PROFINET IO alarms, 241
PROFINET IO performance, 264
PROFINET IO-Device scan, 233
ring topology, 245
sample logic for Modbus TCP client, 197
user defined types, 132
External blocks, 102
Fault contacts, 145
Fault handling
actions, 137
CPU configuration, 44
overview, 136
system, 142
system response, 136
Fault references
alarm contacts, 146
point faults, 146
Fault table, 212
Fault tables
using, 138
Fault tables, Controller, 138
descriptions and corrective actions, 150, 156, 241
groups, 147
Fault tables, I/O, 140
categories, 159
descriptions and corrective actions, 161
groups, 158
Faults
fault contacts, 145
system, 142
non-configurable, 144
user-defined, 139
%SA bits, 126
FCC notice, 271
Flash memory, 81
Floating point numbers, 130
errors in, 131
Formal parameters
restrictions, 97

FT Output of the COMM_REQ function block, 176, 177
Function Block Diagram, 106
Gateway
PNC, 53
configuration, 55
Genius I/O
diagnostic data collection, 89
Getting started, 32
Gigabit Ethernet (GbE)
connectors, 27
Gigabit Ethernet (GbE) interface
configuring, 68
features, 169
operation, 169
status bits, 215
Global data references (%G), 121
Glossary
PROFINET terms, 225
Gratuitous ARP
PROFINET, defined, 225
Grounding, 18
GSDML
PROFINET, defined, 225
GSDML file
third-party devices, 59, 224
Hardware variables, 113
coupled, 114
Humidity specification, 14
I/O scanning, 232
I/O system
initialization, 90
I/O variables, 113
coupled, 114
Indicators
ports, 16
power, 15
Indirect references
word, 119
Input references (%I), 121
Inputs default, 238
Installation, 17
directly on a panel, 21
guidelines, 17
on a DIN rail, 22
space required, 20
Instruction set
by functional group, 107
function block timing, 256
operands, LD, 134
Intelligent Display Module (IDM)
assigning temporary IP address, 33
controller communications window timer, 39
initial powerup, 34
installing, 25
overview, 36
part number, 12
passwords, 36, 87
setting run/stop mode, 80
setting Sweep modes and Communications Window values, 41
Internal references (%M), 121
Interrupt blocks, 109
interrupt handling, 109
scheduling, 111
timed interrupts, 110
IO CRs, 228
IO Devices, 222
IOC
PROFINET, defined, 225
IOC (I/O controller), 143
IOCR
PROFINET, defined, 225
IOCS
PROFINET, defined, 225
IOD
PROFINET, defined, 225
IO-Devices
properties, 63
IOPS
PROFINET
defined, 225
IOxS
PROFINET, defined, 225
IP address, 13
Configuration, 68
Determining if it has been used, 69
duplicate PROFINET controller, 232
duplicate PROFINET device, 231
Format, 170
Isolated network, 68
PNC, 53

PNC LAN configuration, 54
PROFINET Controller, 52
Ladder Diagram, 106
Ladder programming
COMM_REQs, 196
LAN
specifications, 13
LAN configuration, 50
ID, 54
IP address, 54
Name, 54
Properties, 54
LAN Interface OK bit, 216
LAN Interface Status (LIS) bits, 215
LAN Interface Status bits, 176
LAN View, 50, 54
Last scans, 40, 79
LDPROG01, 91
LEDs
ports, 16
power, 15
Limits
data loading, 48
LLDP (Link Layer Discover Protocol), 253
PROFINET, defined, 225
Logic program
controlling execution of COMM_REQs, 176
Logic/configuration power-up source, 82
Loss of device, 161, 238
Loss of IO Controller, 161
LREAL numbers
internal format of, 130
MAC addresses, 13
Mapping
modbus to ENIU memory, 171
Media Redundancy
configuration, 51
Media Redundancy Clients (MRCs), 242
Media Redundancy Manager (MRM), 242
Media Redundancy Protocol (MRP), 242
Memory, 13
configuration, 42
retention of data memory across power cycle, 90
MIB (management information base), 249
MIB-II groups supported, 251
Modbus
reference tables, 171
Modbus Address Space Mapping, 173
Modbus Function Codes, 174
Modbus TCP Channel Commands, 180
Establish Write Channel (2004), 189
Establishing a channel (3001), 182
Mask Write Register Request to a Modbus Server Device (3009), 195
Open a Modbus TCP Client Connection (3000), 180
Read Data from a Modbus TCP Device (3003, 183
Read/Write Multiple Registers to/from a Modbus Server Device (3005), 193
Modbus TCP Channel Commands, 175
Mode transition
stop-to-run, 80
Modes of operation, CPU, 78
Monitoring the communications channel, 203
Mounting
dimensions, 20
DIN rail, 22
orientation, 19
panel, direct, 21
panel, with a Backplate, 24
space required, 20
MRC
PROFINET, defined, 225
MRM
PROFINET, defined, 225
MRP
PROFINET, defined, 225
MRP failover performance, 243
Multicast
PROFINET, defined, 225
NaN (Not a Number)
defined, 131
Nested calls, 92
Network speed
configuration, 54
Network utilization, 54
NMS (network management system), 249
PROFINET, defined, 225
Non-volatile configuration parameters, 53
Normal block scheduling, 111

Normal sweep mode
application program task execution, 73
programmer communications window, 73
Null system configuration for RUN mode, 154
Numerical data, 128
OEM protection, 88
Online editing, 156
Operands for instructions, 134
Operation overview
PROFINET, 227
Output references (%Q), 121
Output scan, 73
Outputs default, 238
Overflow
floating point numbers, 131
Overrides, 123
Parameter passing mechanisms, 105
Parameterized blocks, 95
and local data, 95
reference out of range, 95
referencing formal parameters, 96
Passwords, 87
enabling after disabled, 88
Performance
CPU, 256
PROFINET
examples, 264
factors, 234
Phase
PROFINET, defined, 225
Ping
pinging the TCP/IP interfaces on the network, 69
restrictions, 218
PNIO_CONTROLLER_REF Variable, 235
PNIO_DEV_COMM function block, 236
PNIO_DEVICE_REF Variable, 235
Point faults, 146
Ports, 221
Gigabit Ethernet (GbE), 27
PROFINET, 29
summary, 26
Power supply requirements, 26
Power-down sequence, 90
Powerup, 239
Power-up sequence, 90
CPU memory validation, 90
I/O system initialization, 90
logic/configuration source, 82
power-up self-test, 90
system configuration, 90
Preemptive block scheduling, 111
Privilege levels, 87
PROFINET
connectors, 29
PROFINET communications, 229
PROFINET Controller (PNC)
configuration, 49
diagnostics, 239
parameters, 50
status bits, 240
PROFINET IO alarms, 241
Program blocks, 94
and local data, 94, 104
how blocks are called, 92
types, 93
Program execution
controlling, 109
Program name, 91
Program organization, 91
Program register references (%P), 119
Program scan, 73
Program structure
how blocks are called, 92
program blocks and local data, 104
Programmer Response, 219
Protection level request, 87
RDO
PROFINET, defined, 225
REAL numbers
internal format of, 130
Real time clock (RTC), 83
battery, replacing, 30
Record Data CRs, 228
Reduction Ratio
PROFINET, defined, 226
Redundant Media, PNC
monitor count, 53
Ring Port, 53
Role, 53
test interval, 53
Reference ID Variables (RIVs), 52, 235

References, 119
associated transitions and overrides, 123
data scope, 124
discrete references, 121
indirect, 119
register references, 119
size and default value, 122
system fault references, 142
system status (%S), 125
Remote Node
PROFINET, defined, 226
Retentiveness
of logic and data across power cycle, 90
of logic and data across Stop to Run mode transition, 123
RFI standards, 271
Ring port
configuration, 51
Ring topologies, 242
with multiple controllers, 246
with one controller, 245
RTA
PROFINET, defined, 226
RTC (Real-Time Cyclic)
PROFINET, defined, 226
Run/stop operations, 78
run/outputs disabled, 78
run/outputs enabled, 78
stop/IO scan, 78
stop/No IO scan, 78
Scan parameters, 39
Scan set
for remote I/O modules, 238
Scan sets
parameters, 44
Scanning
I/O, 232
Scope
data, 124
Security, system, 87
privilege levels, 87
Self-test
I/O system initialization, 90
power-up self-test, 90
Send Clock
PROFINET, defined, 226
Send Offset
PROFINET, defined, 226
Sequencing communications requests, 203
Server Capability, 169
Server Protocol Services, 171
Service Request Transfer Protocol (SRTP) inactivity timeout, 170, 219
Shock specification, 14
Simple isolated network configuration, 68
SNMP (Simple Network Management Protocol), 249
features supported, 250
PROFINET, defined, 226
Station Manager, 169
Ethernet interface diagnostics, 214
Station Manager supported by Modbus Server, 171
Status address location, 50, 69
Status bits
CPU, 13
Gigabit Ethernet (GbE) interface, 215
PROFINET Controller, 240
Status data, Channel Commands, 176
STOP modes, 79
Storing configuration, 70
Structure of application programs, 91
Structure variables, 99
Structured Text, 106
Submodule
PROFINET, defined, 226
Subnet mask
PNC, 53
defined, 55
Subroutines
Call function, 109
Sweep, CPU, 71
modes, 74
Stop modes, 79
Symbolic variables, 113
System configuration, 90
System fault references, 142
System operation
clocks and timers, 83
I/O system, 89
passwords, 87
power-down sequence, 90

power-up sequence, 90
retention of data memory across power cycle, 90
system security, 87
System register references (%R), 119
System status references (%S), 121, 125
Technical Support, 3
Temperature, operating, 14
Temporary references (%T), 121
Test packet timeout interval
media redundancy, 244
Third-party devices
configuration, 59
Third-party IO Devices, 224
Time tick references, 125
Timed contacts, 86, 125
Timed interrupts, 110
Time-of-day clock, 83
reading and setting, 83
Timeout Errors, 218
Timers, 83
function blocks, updated, 256
watchdog timer, 84
Timing
I/O scanning, 232
instructions, 256
Transitions, 123
Troubleshooting
Ladder programs, 202
UDFBs
defining, 98
instance data, 99
instances, 99
internal variables, 101
logic restrictions, 101
parameters, 100
scope, 99
Unicast
PROFINET, defined, 226
Update rate (update period)
PROFINET, defined, 226
Update rate (update period), PROFINET
available rates, 13
configuring, 63, 234
data loading, 48
in MRP ring, 243
User defined faults, 139
User defined types (UDTs), 132
User references, 119
system fault references, 142
Variables, 112
C, initialization, 103
I/O, 113
coupled, 114
mapped, 112
member, 98
symbolic, 113
VersaMax modules
configuration, 58
VersaMax PROFINET Scanner
parameters, 57
Vibration specification, 14
VLAN Priority Settings, 230
Watchdog timer
restarting, 84
Watchdog timers, 84
hardware, 85
software, 84
Window modes, 77
Constant Window mode, 77
Limited mode, 77
Run-to-Completion, 77
Word references, 119
Word register references (%W), 119
Word-for-word changes
attempting to correct parameterized block reference, 95
defined, 135
privilege level, 87
symbolic variables, 135
Y0 parameter, 94, 96

GE Intelligent Platforms Information Centers

Headquarters:
1-800-433-2682 or 1-434-978-5100

Global regional phone numbers are available on our web site
www.ge-ip.com

Additional Resources

For more information, please visit the GE Intelligent Platforms web site:

www.ge-ip.com

©2012 GE Intelligent Platforms, Inc. All Rights Reserved

***Trademark of GE Intelligent Platforms, Inc.**

All other brands or names are property of their respective holders.

GFK-2816